Scaling of differential equations

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Aug 14, 2015

Preface

Finding proper values of physical parameters in mathematical models is often quite a challenge. While many has gotten away with using just the mathematical symbols when doing science and engineering with pen and paper, the modern world of numerical computing requires each physical parameter to have a numerical value, otherwise one cannot get started with the computations. For example, in the simplest possible transient heat conduction simulation, a case relevant for a real physical material needs values for the heat capacity, the density, and the heat conduction coefficient of the material. In addition, relevant values must be chosen for initial and boundary temperatures as well as the size of the material. With a dimensionless mathematical model, as explained in Section 5, *no physical quantities* need to be assigned (!). Not only is this a simplification of great convenience, as one simulation is valid for any type of material, but it also actually increases the understanding of the physical problem.

Scaling of differential equations is basically a simple mathematical process, consisting of the chain rule for differentiation and some algebra. The *choice of scales*, however, is a non-trivial topic, which may cause confusion among practitioners without extensive experience with scaling. How to choose scales is unfortunately not well treated in the literature. Most of the times, authors just state scales without proper motivation. The choice of scales is highly problem-dependent and requires knowledge of the characteristic features of the solution or the physics of the problem. The present document aims at explaining "all nuts and bolts" of the scaling technique, including choice of scales, the algebra, the interpretation of dimensionless parameters in scaled models, and how scaling impacts software for solving differential equations.

Traditionally, scaling was mainly used to identify small parameters in mathematical models, such that perturbation methods based on series expansions in terms of the small parameters could be used as an approximate solution method for differential equations. Nowadays, the greatest practical benefit of scaling is related to running numerical simulations, since scaling greatly simplifies the choice of values for the input data and makes the simulations results more widely applicable. The number of parameters in scaled models may be much less than the number of physical parameters in the original model. The parameters in scaled models are also dimensionless and express *ratios* of physical effects rather than the levels of individual effects. Setting meaningful values of a few dimensionless numbers is much easier than determining physically relevant values for the original physical parameters.

Another great benefit of scaling is the physical insight that follows from dimensionless parameters. Since physical effects enter the problem through a few dimensionless groups, one

can from these groups see how different effects compete in their impact on the solution. Ideally, a good physical understanding should provide the same insight, but it is not always easy to "think right" and realize how spatial and temporal scales interact with physical parameters. This interaction becomes clear through the dimensionless numbers, and such numbers are therefore a great help, especially for students, in developing a correct physical understanding.

Since we have a special focus on scaling related to numerical simulations, the document contains a lot of examples on how to program with dimensionless differential equation models. Most numerical models feature quantities with dimension, so we show in particular how to utilize such existing models to solve the equations in the associated scaled model.

Scaling is not a universal mathematical technique as the details depend on the problem at hand. We therefore present scaling in a range of specific applications, starting with simple ODEs, progressing with basic PDEs, before attacking more complicated models, especially from fluid mechanics.

Section 1 discusses units and how to make programs that can automatically take care of unit conversion (the most frequent mathematical mistake in industry and science?). Section 2 introduces the mathematics of scaling and the thinking about scales in a simple ODE problems modeling exponential decay. The ideas are generalized to nonlinear ODEs and to systems of ODEs. Another ODE example on mechanical vibrations is treated in Section 3, where we cover many different physical contexts and different choices of scales. Scaling the standard, linear wave equation is the topic of Section 4, with discussion of how boundary and initial conditions can influence the choice of scales. Another PDE example, the diffusion equation, appears in Section 5. Here we progress from a simple linear diffusion equation in 1D to the impact on scales of an oscillating boundary condition and to nonlinear diffusion models. The final sections are devoted to many famous PDEs arising from continuum models: elasticity, viscous fluid flow, thermal convection, etc.

Especially experimental fluid mechanics is full of relations involving dimensionless numbers such as the Grashof number and the Prandtl number, but none of the textbooks the author has seen explain how these numbers actually relate to dimensionless forms of the governing equations. Consequently, this non-trivial topic is particularly highlighted in our fluid mechanics examples.

To conclude, the purpose of this document is to demystify the technique and thinking about scaling differential equations and motivate for always bringing a differential equation model on dimensionless form before doing numerical simulations.

Contents

List of Exercises and Problems

Scaling is an extremely useful technique in mathematical modeling and numerical simulation. The purpose of the technique is three-fold:

1. Make independent and dependent variables dimensionless.
2. Make the size of independent and dependent variables about unity.
3. Reduce the number of independent physical parameters in the model.

The first and second item mean that for any variable, denote it by q, we introduce a corresponding dimensionless variable

$$\bar{q} = \frac{q - q_0}{q_c},$$

where q_0 is a reference value of q ($q_0 = 0$ is a common choice) and q_c is a characteristic size of $|q|$. Since the numerator and denominator have the same dimension, $\bar{q}$ becomes a dimensionless number.

If q_c is the maximum value of $|q - q_0|$, we see that $0 < |\bar{q}| \leq 1$. How to find q_c is sometimes the big challenging of scaling. Examples will illustrate various approaches to meet this challenge.

The forthcoming text has the following goals.

- Teach the technical steps of making a mathematical model, based on differential equations, dimensionless.
- Describe various techniques for reasoning about the scales, i.e., finding the characteristic sizes of quantities.
- Teach how to identify and interpret dimensionless numbers arising from the scaling process.
- Provide a lot of different examples on making models dimensionless with physically correct scales.
- Demonstrate software tools for computing with numbers with units, including doing unit conversions.
- Introduce software tools for creating user interfaces that can automatically perform unit conversion.
- Use symbolic software (SymPy) to derive exact solutions of differential equations.
- Explain how to run a dimensionless model with software developed for the problem with dimensions.

Limited scope.

Literature covering scaling and non-dimensionalization often also discuss topics such as deriving functional relationships based on dimensional reasoning, as well as dynamic similarity and use of dimensionless groups in experimental investigations [1, 7]. These topics are not covered herein, because our focus is strictly on making models based on differential equations dimensionless.

1 Dimensions and units

A mechanical system undergoing one-dimensional damped vibrations can be modeled by the equation

$$mu'' + bu' + ku = 0, \tag{1}$$

where m is the mass of the system, b is some damping coefficient, k is a spring constant, and $u(t)$ is the displacement of the system. This is an equation expressing the balance of three physical effects: mu'' (mass times acceleration), bu' (damping force), and ku (spring force). The different physical quantities, such as m, $u(t)$, b, and k, have all different *dimensions*, measured in different *units*, but mu'', bu', and ku must all have the same dimension, otherwise it would not make sense to add them (you will have the same problem as when trying to a banana, an apple, and an orange).

1.1 Base units and dimensions

There are seven fundamental *base units*: length, mass, time, electric current, thermodynamic temperature, amount of substance, and luminous intensity. The base units for length, mass, and time, which we are mostly occupied with in this document, are measured by the SI units meter (m), kilogram (kg), and seconds (s), respectively.

The dimension of length is written as [L], the dimension of mass is [M], the dimension of time is [T], and the dimension of temperature is [Θ] (the dimensions of the other base units are not of interest here). The dimension of a *derived unit* like velocity, which is distance (length) divided by time, then becomes $[\mathrm{LT}^{-1}]$. The dimension of force, another derived unit, is the same as the dimension of mass times acceleration, and hence the dimension of force is $[\mathrm{MLT}^{-2}]$.

Let us find the dimensions of the terms in (1). A displacement $u(t)$ has dimension [L]. The derivative $u'(t)$ is change of displacement, which has dimension [L], divided by a time interval, which has dimension [T], implying that the dimension of u' is LT^{-1}. This results coincides with the interpretation of u' as velocity and the fact that velocity is defined as distance ([L]) per time ([T]).

Looking at (1), and interpreting $u(t)$ as displacement, we realize that the term mu'' (mass times acceleration) has dimension $[\mathrm{MLT}^{-2}]$. The term bu' must have the same dimension, and since u' has dimension $[\mathrm{LT}^{-1}]$, b must have dimension $[\mathrm{MT}^{-1}]$. Finally, ku must also have dimension $[\mathrm{MLT}^{-1}]$, implying that k is a parameter with dimension $[\mathrm{MT}^{-2}]$.

The unit of a physical quantity follows from the dimension expression. For example, since velocity has dimension LT^{-1} and length is measured in m while time is measured in s, the unit for velocity becomes m/s. Similarly, force has dimension $[\mathrm{MLT}^{-2}]$ and unit kg $\mathrm{m/s}^2$. The k parameter in (1) is measured in $\mathrm{kg\,s}^{-2}$.

Many derived quantities are measured in derived units. Force is one example: Newton (N) is a derived unit for force, equal to kg $\mathrm{m/s}^2$. Another derived unit is Pascal (Pa) for pressure and stress, i.e., force per area. The unit of Pa then equals $\mathrm{N/m}^2$ or $\mathrm{kg/ms}^2$. Below are more derived quantities and their units.

Name	Symbol	Physical quantity	unit
radian	rad	angle	1
hertz	Hz	frequency	s^{-1}
newton	N	force, weight	kg m/s^2
pascal	Pa	pressure, stress	N/m^2
joule	J	energy, work, heat	Nm
watt	W	power	J/s

Some common physical quantities and their dimensions are listed next.

Quantity	relation	unit	dimension
stress	force/area	$N/m^2 = Pa$	$[MLT^{-2}L^{-2}]$
pressure	force/area	$N/m^2 = Pa$	$[MLT^{-2}L^{-2}]$
density	mass/volume	kg/m^3	$[ML^{-3}]$
strain	displacement/length	1	[1]
Young's modulus	stress/strain	$N/m^2 = Pa$	$[MLT^{-2}L^{-2}]$
Poisson's ratio	transverse strain/axial strain	1	[1]
moment (of a force)	force $\times$ distance	Nm	$[ML^2T^{-2}]$
impulse	force $\times$ time	Ns	$[MLT^{-1}]$
linear momentum	force $\times$ velocity	Nm/s	$[ML^2T^{-3}]$
work	force $\times$ distance	$Nm = J$	$[ML^2T^{-2}]$
energy	work	$Nm = J$	$[ML^2T^{-2}]$
power	work/time	$Nm/s = W$	$[ML^2T^{-3}]$
heat	work	J	$[ML^2T^{-2}]$
heat flux	heat rate/area	Wm^{-2}	$[MT^{-3}]$
temperature	base unit	K	$[\Theta]$
heat capacity	heat change/temperature change	J/K	$[ML^2T^{-2}\Theta^{-1}]$
specific heat capacity	heat capacity/unit mass	$JK^{-1}kg^{-1}$	$[L^2T^{-2}\Theta^{-1}]$
thermal conductivity	heat flux/temperature gradient	$Wm^{-1}K^{-1}$	$[MLT^{-3}\Theta^{-1}]$
dynamic viscosity	shear stress/velocity gradient	$kgm^{-1}s^{-1}$	$[ML^{-1}T^{-1}]$
kinematic viscosity	dynamic viscosity/density	$[m^2/s$	$L^2T^{-1}]$
surface tension	energy/area	$[J/m^2$	$MT^{-2}]$

Prefixes. Units often have prefixes. For example, kilo is a prefix for 1000, so kg is 1000 g. Similarly, GPa means giga pascal or 10^9 Pa.

Mathematically, it does not matter what units we use for a physical quantity. However, when we deal with approximations and errors, units are important. Suppose we work with a geophysical problem where the length scale is typically measured in km and we have an approximation 12.5 km to the exact value 12.52 km. The error is then 0.02 km. Switching units to mm leads to an error of 20,000 mm. A program working in mm would report $2 \cdot 10^5$ as the error, while a program working in km would print 0.02. The absolute error is therefore sensitive to the choice of units. This fact motivates for the use of the *relative error*: (exact - approximate)/exact since then the unit cancels. In the present example, one gets a relative error of $1.6 \cdot 10^{-3}$ regardless of the whether the length is measured in km or mm.

Nevertheless, rather than relying solely on relative errors, it is in general better to scale the problem such that the quantities entering the computations are of unit size (or at least moderate) instead of being very large or very small. The techniques of this document show how this can be done.

SI vs US/British systems. Confusion arises quickly when some physical quantities are expressed in SI units while others are in US or British units. Density could, for instance, be given in unit of ounce per teaspoon (see Exercise 1 for how to safely convert to a standard unit like $\mathrm{kg\,m^{-3}}$). Although unit conversion is a mathematical topic much trained in school, errors in conversions between units probably rank highest among all errors committed by engineers. Having good software tools to assist in unit conversion is therefore paramount, and this topic is treated next. Readers who are primarily interested in the mathematical scaling technique may safely jump over Sections 1.2 and 1.3 and continue with Section 2.

1.2 PhysicalQuantity: object for computing with units

Computations with units are supported by the `PhysicalQuantity` object from the ScientificPython package[1] by Konrad Hinsen. Unfortunately, ScientificPython does not, at the time of this writing, work with NumPy version 1.9 or later, so we have isolated the `PhysicalQuantity` object in a module `PhysicalQuantities`[2]. There is also an alternative package Unum[3] for computing with numbers with units, but we shall stick to the former module here.

Let us demonstrate the usage of the `PhysicalQuantity` object by computing $s = vt$, where v is a velocity given with unit yards per minute and t is time measured in hours. Run `pydoc PhysicalQuantities`, or

```
Terminal> pydoc Scientific.Physics.PhysicalQuantities
```

if you have ScientificPython installed, to see what the name of the units are. Yards are specified by `yd`, minutes by `min`, and hours by `h`. We can now compute $s = vt$ as follows:

```
>>> # With ScientificPython:
>>> from Scientific.Physics.PhysicalQuantities import \
... PhysicalQuantity as PQ
>>> # With PhysicalQuantities as separate/stand-alone module:
>>> from PhysicalQuantities import PhysicalQuantity as PQ
>>>
>>> v = PQ('120 yd/min')   # velocity
>>> t = PQ('1 h')          # time
>>> s = v*t                # distance
>>> print s                # s is string
120.0 h*yd/min
```

The odd unit `h*yd/min` is better converted to a standard SI unit such as meter:

```
>>> s.convertToUnit('m')
>>> print s
6583.68 m
```

Note that `s` is a `PhysicalQuantity` object with a value and a unit. For mathematical computations we need to extract the value as a `float` object. We can also extract the unit as a string:

```
>>> print s.getValue()     # float
6583.68
>>> print s.getUnitName()  # string
m
```

[1] https://bitbucket.org/khinsen/scientificpython
[2] https://github.com/hplgit/physical-quantities
[3] https://bitbucket.org/kiv/unum/

Here is an example on how to convert the odd velocity unit yards per minute to something more standard:

```
>>> v.convertToUnit('km/h')
>>> print v
6.58368 km/h
>>> v.convertToUnit('m/s')
>>> print v
1.8288 m/s
```

As another example on unit conversion, say you look up the specific heat capacity of water to be 1 $\mathrm{cal\,g^{-1}K^{-1}}$. What is the corresponding value in the standard unit $\mathrm{Jg^{-1}K^{-1}}$ where joule replaces calorie?

```
>>> c = PQ('1 cal/(g*K)')
>>> c.convertToUnit('J/(g*K)')
>>> print c
4.184 J/K/g
```

1.3 Parampool: user interfaces with automatic unit conversion

The Parampool[4] package allows creation of user interfaces with support for units and unit conversion. Values of parameters can be set as a number with a unit. The parameters can be registered beforehand with a preferred unit, and whatever the user prescribes, the value and unit are converted so the unit becomes the registered unit. Parampool supports various type of user interfaces: command-line arguments (option-value pairs), text files, and interactive web pages. All of these are described next.

Example application. As case, we want to make software for computing with the simple formula $s = v_0 t + \frac{1}{2}at^2$. We want v_0 to be a velocity with unit m/s, a to acceleration with unit $\mathrm{m/s^2}$, t to be time measured in s, and consequently s is a distance measured in m.

Pool. First, Parampool requires us to define a *pool* of all input parameters, which is here simply represented by list of dictionaries, where each dictionary holds information about one parameter. It is possible to organize input parameters in a tree structure with subpools having subpools, but for our simple application we just need a flat structure with three input parameters: v_0, a, and t. These parameters are put in a subpool called "Main". The pool is created by the code

```
def define_input():
    pool = [
        'Main', [
            dict(name='initial velocity', default=1.0, unit='m/s'),
            dict(name='acceleration', default=1.0, unit='m/s**2'),
            dict(name='time', default=10.0, unit='s')
            ]
        ]

    from parampool.pool.UI import listtree2Pool
    pool = listtree2Pool(pool)  # convert list to Pool object
    return pool
```

[4] https://github.com/hplgit/parampool

For each parameter we can define a logical name, such as `initial velocity`, a default value, and a unit. Additional properties are also allowed, see the Parampool documentation[5].

Tip: specify default values of numbers as float objects.

Note that we write 1.0 as default value and not just 1. In the latter case, Parampool will interpret that our parameter is an integer and actually convert input like `2.5 m/s` to `2 m/s`. To ensure that a real-valued parameter becomes a `float` object inside the pool, specify the default value as a real number: `1.` or `1.0`. (The type of an input parameter can alternatively be explicitly set by the `str2type` property, e.g., `str2type=float`.)

Fetching pool data and computing s. We can make a little function for fetching values from the pool and computing s:

```
def distance(pool):
    v_0 = pool.get_value('initial velocity')
    a = pool.get_value('acceleration')
    t = pool.get_value('time')
    s = v_0*t + 0.5*a*t**2
    return s
```

The `pool.get_value` function returns the numerical value of the named parameter, after the unit has been converted from what the user has specified to what was registered in the pool. For example, if the user provides the command-line argument `-time '2 h'`, Parampool will convert this quantity to seconds and `pool.get_value('time')` will return 7200.

Reading command-line options. To run the computations, we define the pool, load values from the command line, and call `distance`:

```
pool = define_input()
from parampool.menu.UI import set_values_from_command_line
pool = set_values_from_command_line(pool)

s = distance(pool)
print 's=%g' % s
```

Parameter names with whitespace must use an underscore for whitespace in the command-line option, such as in `--Initial_velocity`. We can now run

```
Terminal> python distance.py --initial_velocity '10 km/h' \
          --acceleration 0 --time '1 h
s=10000
```

Notice from the answer (`s`) that 10 km/h gets converted to m/s and 1 h to s.

It is also possible to fetch parameter values as `PhysicalQuantity` objects from the pool through calling

[5] `http://hplgit.github.io/parampool/doc/web/index.html`

```
v_0 = pool.get_value_unit('Initial velocity')
```

The following variant of the `distance` function computes with values and units:

```
def distance_unit(pool):
    """Compute distance $s = v_0t + \frac{1}{2}at^2$. (DocOnce)"""
    # Compute with units
    from parampool.PhysicalQuantities import PhysicalQuantity as PQ
    v_0 = pool.get_value_unit('initial velocity')
    a = pool.get_value_unit('acceleration')
    t = pool.get_value_unit('time')
    s = v_0*t + 0.5*a*t**2
    return s.getValue(), s.getUnitName()
```

We can then do

```
s, s_unit = distance_unit(pool)
print 's=%g' % s, s_unit
```

and get output with the right unit as well.

Setting default values in a file. In large applications with lots of input parameters one will often like to define a (huge) set of default values specific for a case and then override a few of them on the command-line. Such sets of default values can be set in a file using syntax like

```
subpool Main
initial velocity = 100 ! yd/min
acceleration = 0 ! m/s**2         # drop acceleration
end
```

The unit can be given after the `!` symbol (and before the comment symbol `#`).

To read such files we have to add the lines

```
from parampool.pool.UI import set_defaults_from_file
pool = set_defaults_from_file(pool)
```

before the call to `set_defaults_from_command_line`.

If the above commands are stored in a file `distance.dat`, we give this file information to the program through the option `-poolfile distance.dat`. Running just

```
Terminal> python distance.py --poolfile distance.dat
s=15.25 m
```

first loads the velocity 100 yd/min converted to 1.524 m/s and zero acceleration into the pool system and, and then we call `distance_unit`, which loads these values from the pool along with the default value for time, set as 10 s. The calculation is then $s = 1.524 \cdot 10 + 0 = 15.24$ with unit m. We can override the time and/or the other two parameters on the command line:

```
Terminal> python distance.py --poolfile distance.dat --time '2 h'
s=10972.8 m
```

The resulting calculations are $s = 1.524 \cdot 7200 + 0 = 10972.8$. You are encouraged to play around with the `distance.py`[6] program.

[6] `http://tinyurl.com/nm5587k/scale/distance.py`

Specifying multiple values of input parameters. Parampool has an interesting feature: multiple values can be assigned to an input parameter, thereby making it easy for an application to run through all combinations of all parameters. We can demonstrate this feature by making a table of v_0, a, t, and s values. In the compute function, we need to call `pool.get_values` instead of `pool.get_value` to get a list of all the values that were specified for the parameter in question. By nesting loops over all parameters, we visit all combinations of all parameters as specified by the user:

```
def distance_table(pool):
    """Grab multiple values of parameters from the pool."""
    table = []
    for v_0 in pool.get_values('initial velocity'):
        for a in pool.get_values('acceleration'):
            for t in pool.get_values('time'):
                s = v_0*t + 0.5*a*t**2
                table.append((v_0, a, t, s))
    return table
```

In case just a single value was specified for a parameter, `pool.get_values` returns this value only and there will be only one pass in the associated loop.

After loading command-line arguments into our `pool` object, we can call `distance_table` instead of `distance` or `distance_unit` and write out a nicely formatted table of results:

```
table = distance_table(pool)
print '|-----------------------------------------------------|'
print '|      v_0   |      a     |      t     |      s       |'
print '|-----------------------------------------------------|'
for v_0, a, t, s in table:
    print '|%11.3f | %10.3f | %10.3f | %12.3f |' % (v_0, a, t, s)
print '|-----------------------------------------------------|'
```

Here is a sample run,

```
Terminal> python distance.py --time '1 h & 2 h & 3 h' \
          --acceleration '0 m/s**2 & 1 m/s**2 & 1 yd/s**2' \
          --initial_velocity '1 & 5'
|-----------------------------------------------------|
|      v_0   |      a     |      t     |      s       |
|-----------------------------------------------------|
|      1.000 |      0.000 |   3600.000 |     3600.000 |
|      1.000 |      0.000 |   7200.000 |     7200.000 |
|      1.000 |      0.000 |  10800.000 |    10800.000 |
|      1.000 |      1.000 |   3600.000 |  6483600.000 |
|      1.000 |      1.000 |   7200.000 | 25927200.000 |
|      1.000 |      1.000 |  10800.000 | 58330800.000 |
|      1.000 |      0.914 |   3600.000 |  5928912.000 |
|      1.000 |      0.914 |   7200.000 | 23708448.000 |
|      1.000 |      0.914 |  10800.000 | 53338608.000 |
|      5.000 |      0.000 |   3600.000 |    18000.000 |
|      5.000 |      0.000 |   7200.000 |    36000.000 |
|      5.000 |      0.000 |  10800.000 |    54000.000 |
|      5.000 |      1.000 |   3600.000 |  6498000.000 |
|      5.000 |      1.000 |   7200.000 | 25956000.000 |
|      5.000 |      1.000 |  10800.000 | 58374000.000 |
|      5.000 |      0.914 |   3600.000 |  5943312.000 |
|      5.000 |      0.914 |   7200.000 | 23737248.000 |
|      5.000 |      0.914 |  10800.000 | 53381808.000 |
|-----------------------------------------------------|
```

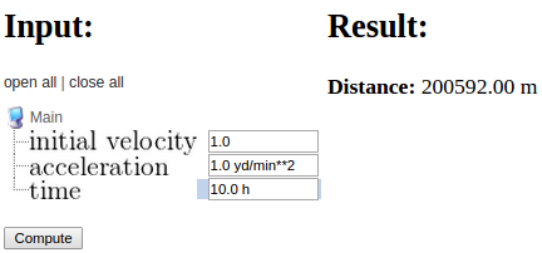

Figure 1: Web GUI where parameters can be specified with units.

Notice that some of the multiple values have dimensions different from the registered dimension for that parameter, and the table shows that conversion to the right dimension has taken place.

Generating a graphical user interface. For the fun of it, we can easily generate a graphical user interface via Parampool. We wrap the `distance_unit` function in a function that returns the result is some nice-looking HTML code:

```
def distance_unit2(pool):
    # Wrap result from distance_unit in HTML
    s, s_unit = distance_unit(pool)
    return '<b>Distance:</b> %.2f %s' % (s, s_unit)
```

In addition, we must make a file `generate_distance_GUI.py` with the simple content

```
from parampool.generator.flask import generate
from distance import distance_unit2, define_input

generate(distance_unit2, pool_function=define_input, MathJax=True)
```

Running `generate_distance_GUI.py` creates a Flask-based web interface[7] to our `distance_unit` function, see Figure 1. The text fields in this GUI allow specification of parameters with numbers and units, e.g., acceleration with unit yards per minute squared, as shown in the figure. Hovering the mouse slightly to the left of the text field causes a little black window to pop up with the registered unit of that parameter.

With examples shown above, you should be able to utilize the `PhysicalQuantity` object and the Parampool package in your programs to work safely with units. For the coming text, where we discuss the craft of scaling in detail, we shall just work in standard SI units and avoid unit conversion so there will be no more use of `PhysicalQuantity` and Parampool.

2 Exponential decay

Processes undergoing exponential reduction can be modeled by the ODE problem

$$u'(t) = -au(t), \quad u(0) = I, \tag{2}$$

where $a, I > 0$ are prescribed constants and $u(t)$ is the unknown function. For this particular model, we can easily derive the solution, $u(t) = Ie^{-at}$, which is helpful to have in mind during the scaling process.

[7] You need to have Flask and additional packages installed. This is easy to do with a few `pip install` commands, see [5] or [6].

Example: Population dynamics. The evolution of a population of humans, animals, cells, etc., under unlimited access to resources, can be modeled by (2). Then u is the number of individuals in the population, strictly speaking an integer, but well modeled by a real number in large populations. The parameter a is the increase in the number of individuals per time and per individual.

Example: Decay of pressure with altitude. The simple model (2) also governs the pressure in the atmosphere (under many assumptions). In this case u is the pressure, measured in Nm^{-2}; t is the hight in meters; and $a = M/(R^*T)$, where M is the molar mass of the Earth's air (0.029 kg/mol), R^* is the universal gas constant ($8.314\,\frac{\mathrm{Nm}}{\mathrm{mol\ K}}$), and T is the temperature in Kelvin (K). The temperature depends on the hight so we have $a = a(t)$.

2.1 The technical steps of the scaling procedure

Step 1: Identify independent and dependent variables. There is one independent variable, time t, and one dependent variable, u.

Step 2: Make independent and dependent variables dimensionless. We introduce a new dimensionless t, called $\bar{t}$, defined by

$$\bar{t} = \frac{t}{t_c}, \tag{3}$$

where t_c is a *characteristic value* of t. Similarly, we introduce a dimensionless u, named $\bar{u}$, according to

$$\bar{u} = \frac{u}{u_c}, \tag{4}$$

where u_c is a constant *characteristic size* of u. When u has a specific interpretation, say when (2) models pressure in an atmospheric layer, u_c would be referred to as characteristic pressure. For a decaying population, u may be a characteristic number of members in the population.

Step 3: Derive the model involving only dimensionless variables. The next task is to insert the new dimensionless variables in the governing mathematical model. That is, we replace t by $t_c\bar{t}$ and u by $u_c\bar{u}$ in (2). The derivative with respect to $\bar{t}$ is derived as

$$\frac{du}{dt} = \frac{d(u_c\bar{u})}{d\bar{t}}d\bar{t}dt = u_c\frac{d\bar{u}}{d\bar{t}}\frac{1}{t_c} = \frac{u_c}{t_c}\frac{d\bar{u}}{d\bar{t}}\,.$$

The model (2) now becomes

$$\frac{u_c}{t_c}\frac{d\bar{u}}{d\bar{t}} = -au_c\bar{u}, \quad u_c\bar{u}(0) = I\,. \tag{5}$$

Step 4: Make each term dimensionless. Equation (5) still has terms with dimensions. To make each term dimensionless, we usually divide by the coefficient in front of the term with the highest time derivative (but dividing by any coefficient will do). The result is

$$\frac{d\bar{u}}{d\bar{t}} = -at_c\bar{u}, \quad \bar{u}(0) = u_c^{-1}I\,. \tag{6}$$

Step 5: Estimate the scales. A characteristic quantity like t_c reflects the time scale in the problem. Estimating such a time scale is certainly the most challenging part of the scaling procedure. There are different ways to reason. The first is to aim at a size of $\bar{u}$ and its derivatives that is of order unity. If u_c is chosen such that $|\bar{u}|$ is of size unity, we see from (6) that $d\bar{u}/d\bar{t}$ is of the size of $\bar{u}$ (i.e., unity) if we choose $t_c = 1/a$.

Alternatively, we may look at a special case of the model where we have analytical insight. In the present problem we are lucky to know the exact solution for any value of the input data. For exponential decay, $u(t) \sim e^{-at}$, it is common to define a characteristic time scale t_c as the time it takes to reduce u by a factor of $1/e$ (also called the *e-folding time*):

$$e^{-at_c} = \frac{1}{e}e^{-a\cdot 0} \quad \Rightarrow \quad e^{-at_c} = e^{-1},$$

from which it follows that $t_c = 1/a$.

In this example, two different, yet common ways of reasoning, lead to the same value of t_c. However, instead of using the e-folding time we could use the half-time of the exponential decay as characteristic time, which is also a very common measure of the time scale in such processes. The half time is defined as the time it takes to halve u:

$$e^{-at_c} = \frac{1}{2}e^{-a\cdot 0} \quad \Rightarrow \quad t_c = a^{-1}\ln 2\,.$$

There is a factor $\ln 2 = 0.69$ difference from the other t_c value. As long as the factor is not an order of magnitude or more different, we do not pay attention to such small differences. Although $t_c = a^{-1}\ln 2$ is a fine time scale to be used in this problem, it leads to a scaled differential equation $u' = -(\ln 2)u$, which is fine, but an unusual form. People tend to prefer $u' = -u$, which arises from $t_c = 1/a$. We shall therefore use the latter as time scale.

Regarding u_c, we may look at the initial condition and realize that the choice $u_c = I$ makes $\bar{u}(0) = 1$. For $\bar{t} > 0$ we know that $\bar{u}$ is decreasing, so $u_c = I$ gives us $\bar{u} \leq 1$, which is always a goal. Alternatively, we may look to analytical insight, $u(t) = Ie^{-at}$, to see that $u \leq I$, such that $u_c = I$ gives $\bar{u} \leq 1$.

With $t_c = 1/a$ and $u_c = I$, we have the final dimensionless model

$$\frac{d\bar{u}}{d\bar{t}} = -\bar{u}, \quad \bar{u}(0) = 1\,. \tag{7}$$

This is a remarkable result in the sense that *all physical parameters* (a and I) are removed from the model! Or more precisely, there are no physical input parameters to assign before using the model. In particular, numerical investigations of the original model (2) would need experiments with different a and I values, while numerical investigations of (7) can be limited to *a single run*! As soon as we have computed the curve $\bar{u}(\bar{t})$, we can find the solution $u(t)$ of (2) by

$$u(t) = u_c\bar{u}(t/t_c) = I\bar{u}(at)\,. \tag{8}$$

This particular transformation actually means stretching the $\bar{t}$ and $\bar{u}$ axes in a plot of $\bar{u}(\bar{t})$ by the factors a and I, respectively.

It is very common to drop the bars when the scaled problem has been derived and work further with (7) simply written as

$$\frac{du}{dt} = -u, \quad u(0) = 1\,.$$

2.2 Making software utilizing the dimensionless model

Software for solving (2) could take advantage of the fact that only one simulation of (7) is necessary. As soon as we have $\bar{u}(\bar{t})$ accessible, a simple scaling (8) computes the real $u(t)$ for any given input data a and I. Although the numerical computation of $u(t)$ from (2) is very fast in this simple model problem, using (8) is very much faster than computing a full numerical solution in more complicated differential equation problems.

We can compute with the dimensionless model (7) in two ways, either make a solver for (7) or reuse a solver for (2) with the parameters appropriately set ($I = 1$, $a = 1$). The latter approach has the advantage of giving us software that works both with a dimensionless model and a model with dimensions and all the original physical parameters.

Software for the original problem with dimensions. We base our solver for (7) on a solver for (2). Assume that we have some module `decay.py` that offers the following functions:

- `solver(I, a, T, dt, theta=0.5)` for returning the solution arrays `u` and `t` for (2) solved by the θ rule.
- `read_command_line_argparse()` for reading parameters in the problem from the command line and returning them: `I`, `a`, `T`, `theta` (θ), and a list of Δt values for time steps. (We shall only make use of the first Δt value.)

The basic statements for solving (2) are then

```
from decay import solver, read_command_line_argparse
I, a, T, theta, dt_values = read_command_line_argparse()
u, t = solver(I, a, T, dt_values[0], theta)

from matplotlib.pyplot import plot, show
plot(t, u)
show()
```

The module `decay.py`[8] is developed and explained in Section **??** in [3].

To solve the dimensionless problem, just fix $I = 1$ and $a = 1$:

```
I, a, T, theta, dt_values = read_command_line_argparse()
u, t = solver(I=1, a=1, T=T, dt=dt_values[0], theta=theta)
```

A plain solution. A key observation, as mentioned, is that we need to solve the problem (7) only once. All solutions corresponding to different I and a values in the original physical problem can be recovered by scaling this single solution with formula (8). We therefore want to make software that takes advantage of this fact. When requesting a solution, we see if it has already been computed and stored in a file, and if so, the data can be retrieved from file, otherwise we have to compute a new solution and store it in a file.

The computational recipe goes as follows.

1. A computed solution $\bar{u}(\bar{t})$ is stored in a file with name `u_scaled.dat`.
2. The first line in the file contains T, Δt, and θ used to compute the stored $\bar{u}(\bar{t})$.

[8] `http://tinyurl.com/nm5587k/softeng1/decay.py`

3. The T, Δt, and θ parameters are read from the first line in the file and compared with those required by the user.

4. If one of the three parameters changes, the solution in the file must be recomputed.

The actual code may look as follows:

```
from decay import solver as solver_unscaled
import numpy as np

def solver_scaled(T, dt, theta):
    """
    Solve u'=-u, u(0)=1 for (0,T] with step dt and theta method.
    """
    # Is the scaled problem already solved and dimensionless
    # curve available from file?
    # See if u_scaled.dat has the right parameters.
    already_computed = False
    datafile = 'u_scaled.dat'
    if os.path.isfile(datafile):      # does u_scaled.dat exist?
        infile = open(datafile, 'r')
        infoline = infile.readline()  # read the first line
        words = infoline.split()      # split line into words
        T_, dt_, theta_ = [float(w) for w in words]
        if T_ == T and dt_ == dt and theta_ == theta:
            # The file was computed with the desired data, load
            # the solution into arrays
            data = np.loadtxt(infile)
            u_scaled = data[1,:]
            t_scaled = data[0,:]
            print 'Read scaled solution from file'
            already_computed = True
        infile.close()
    if not already_computed:
        # T, dt or theta is different from u_scaled.dat
        u_scaled, t_scaled = \
           solver_unscaled(I=1, a=1, T=T, dt=dt, theta=theta)
        outfile = open(datafile, 'w')
        outfile.write('%f %f %.1f\n' % (T, dt, theta))
        np.savetxt(outfile, np.array([t_scaled, u_scaled]))
        outfile.close()
        print 'Computed scaled solution'
    return u_scaled, t_scaled

def unscale(u_scaled, t_scaled, I, a):
    return I*u_scaled, a*t_scaled
```

The `np.savetxt` function saves a two-dimensional arrays ("table") to a text file, and the `np.loadtxt` function can load the data back into the program.

Simplifying the implementation with joblib. The Python package `joblib` has functionality that is very convenient for implementing the `solver_scaled` function. The first time a function is called with a set of arguments, the statements in the function are executed and the return value is saved to file. If the function is called again with the same set of arguments, the statements in the function are not executed, but the return value is read from file. In computer science, one would say that `joblib` in this way provides *memorization* functionality for Python functions. This functionality is particularly aimed at large-scale computations with arrays that one would like to avoid being recomputed.

Utilizing `joblib`, our `solver_scaled` function can be dramatically simplified:

```
def solver_scaled(T, dt, theta):
    """
    Solve u'=-u, u(0)=1 for (0,T] with step dt and theta method.
    """
    print 'Computing the numerical solution'
    return solver_unscaled(I=1, a=1, T=T, dt=dt, theta=theta)
```

Then we create some "computer memory on disk", i.e., some disk space to store the result of a call to the `solver_scaled` function. Thereafter, we redefine the name `solver_scaled` to a new function, created by `joblib`, which calls our original `solver_scaled` function if necessary and otherwise loads data from file:

```
import joblib
disk_memory = joblib.Memory(cachedir='temp')
solver_scaled = disk_memory.cache(solver_scaled)
```

The solutions are actually stored in files in the directory `temp`.

A typical use case is to read values from the command line, solve the unscaled problem (if necessary), scale the solution, and visualize the solution with dimension:

```
def main():
    # Read parameters, solve and plot
    I, a, T, theta, dt_values = read_command_line_argparse()
    dt = dt_values[0]  # use only the first dt value
    u_scaled, t_scaled = solver_scaled(T, dt, theta)
    u, t = unscale(u_scaled, t_scaled, I, a)

    plt.figure()
    plt.plot(t_scaled, u_scaled)
    plt.xlabel('scaled time'); plt.ylabel('scaled velocity')
    plt.title('Universial solution of scaled problem')
    plt.savefig('tmp1.png');  plt.savefig('tmp1.pdf')

    plt.figure()
    plt.plot(t, u)
    plt.xlabel('t'); plt.ylabel('u')
    plt.title('I=%g, a=%g, theta=%g' % (I, a, theta))
    plt.savefig('tmp2.png'); plt.savefig('tmp2.pdf')
    plt.show()
```

The complete code resides in the file `decay_scaled.py`[9].

Note that we write a message `Computing the numerical solution` inside the `solver_scaled` function. We can then easily detect when the solution is actually computed and when it is simply read from file. Here is a demo:

```
Terminal> # Very first run
Terminal> python decay_scaled.py --T 7 --a 1 --I 0.5 --dt 0.2
[Memory] Calling __main__--home-hpl...
solver_scaled-alias(7.0, 0.2, 0.5)
Computing the numerical solution

Terminal> # No change of T, dt, theta - can reuse solution in file
Terminal> python decay_scaled.py --T 7 --a 4 --I 2.5 --dt 0.2

Terminal> # Change of dt, must recompute
Terminal> python decay_scaled.py --T 7 --a 4 --I 2.0 --dt 0.5
```

[9] `http://tinyurl.com/nm5587k/scale/decay_scaled.py`

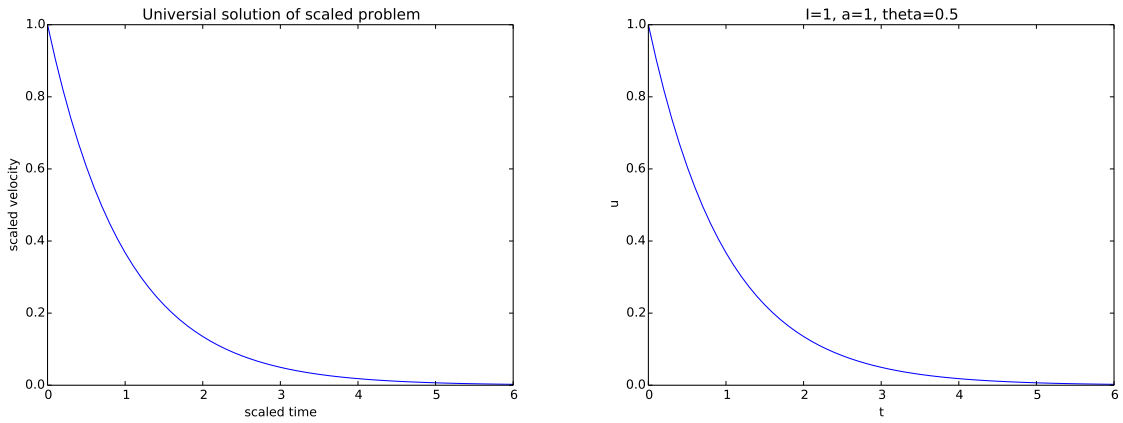

Figure 2: Scaled (left) and unscaled (right) exponential decay.

```
[Memory] Calling __main__--home-hpl...
solver_scaled-alias(7.0, 0.5, 0.5)
Computing the numerical solution

Terminal> # Change of dt again, but dt=0.2 is already in a file
Terminal> python decay_scaled.py --T 7 --a 0.5 --I 1 --dt 0.2
```

We realize that `joblib` has access to all previous runs and does not recompute unless it is strictly required. Our previous implementation without `joblib` used only one file (for one numerical case) and will therefore perform many more calls to `solver_unscaled`.

A plot of the scaled and unscaled solution appears in Figure 2.

On the implementation of a simple memoize function.

A memoized function recalls previous results when the same set of arguments is encountered. That is, the function caches its results. A simple implementation stores the arguments in a function call and the returned results in a dictionary, and if the arguments are seen again, one looks up in the dictionary and return previously computed results:

```
class Memoize:
    def __init__(self, f):
        self.f = f
        self.memo = {}  # map arguments to results

def __call__(self, *args):
        if not args in self.memo:
            self.memo[args] = self.f(*args)
        return self.memo[args]

# Wrap my_compute_function(arg1, arg2, ...)
my_compute_function = Memoize(my_compute_function)
```

The memoize functionality in `joblib.Memory` is more sophisticated and can work very efficiently with large array data structures as arguments. Note that the simple version above

can only be used when all arguments to the function `f` are immutable (since the key in a dictionary has to be immutable).

2.3 Scaling a generalized problem

Now we consider an extension of the exponential decay ODE to the form

$$u'(t) = -au(t) + b, \quad u(0) = I\,. \tag{9}$$

One particular model, with constant a and b, is a spherical micro-organism falling in air,

$$u' = -\frac{3\pi d\mu}{\varrho_b V}u + g\left(\frac{\varrho}{\varrho_b} - 1\right), \tag{10}$$

where d, μ, ϱ_b, ϱ, V, and g are physical parameters. The function $u(t)$ represents the vertical velocity, being positive upwards. We shall use this model in the following.

Exact solution. It can be handy to have the exact solution for reference, in case of constant a and b:

$$u_{\mathrm{e}}(t) = \frac{e^{-at}}{a}\left(b(e^{at} - 1) + aI\right)\,.$$

It can be very handy to use a symbolic computation tool such as SymPy to aid us in solving differential equations. Let us therefore demonstrate how SymPy can be used to find this solution. First we define the parameters in the problem as symbols and $u(t)$ as a function:

```
>>> from sympy import *
>>> t, a, b, I = symbols('t a b I', real=True, positive=True)
>>> u = symbols('u', cls=Function)
```

The next task is to define the differential equation, either as a symbolic expression that is to equal zero, or as an equation `Eq(lhs, rhs)` with `lhs` and `rhs` as expressions for the left- and right-hand side):

```
>>> # Define differential equation
>>> eq = diff(u(t), t) + a*u(t) - b
>>> # or
>>> eq = Eq(diff(u(t), t), -a*u(t) + b)
```

The differential equation can be solved by the `dsolve` function, yielding an equation of the form `u(t) == expression`. We want to grab the expression on the right-hand side as our solution:

```
>>> sol = dsolve(eq, u(t))
>>> print sol
u(t) == (b + exp(a*(C1 - t)))/a
>>> u = sol.rhs                    # grab solution
>>> print u
(b + exp(a*(C1 - t)))/a
```

The solution contains the unknown integration constant `C1`, which must be determined by the initial condition. We form the equation arising from the initial condition $u(0) = I$:

```
>>> C1 = symbols('C1')
>>> eq = Eq(u.subs(t, 0), I)   # substitute t by 0 in u
>>> sol = solve(eq, C1)
>>> print sol
[log(I*a - b)/a]
```

The one solution that was found must then be substituted back in the expression `u` to yield the final solution:

```
>>> u = u.subs(C1, sol[0])
>>> print u
(b + exp(a*(-t + log(I*a - b)/a)))/a
```

As in mathematics with pen and paper, we strive to simplify expressions also in symbolic computing software. This frequently requires some trial and error process with SymPy's simplification functions. A very standard first try is to expand everything and run simplification algorithms:

```
>>> u = simplify(expand(u))
>>> print u
(I*a + b*exp(a*t) - b)*exp(-a*t)/a
```

Note that doing `latex(u)` automatically converts the expression to LaTeX syntax for inclusion in reports.

Theory. The challenges in our scaling is to find the right u_c and t_c scales. From (9) we see that if $u' \to 0$ as $t \to \infty$, u approaches the constant value b/a. It can be convenient to let the scaled $\bar{u} \to 1$ as we approach the $d\bar{u}/d\bar{t} = 0$ state. This idea points to choosing

$$u_c = \frac{b}{a} = g\left(\frac{\varrho}{\varrho_b} - 1\right)\left(\frac{3\pi d\mu}{\varrho_b V}\right)^{-1}. \tag{11}$$

On the sign of the scaled velocity.

A little note on the sign of u_c is necessary here. With $\varrho_b < \varrho$, the buoyancy force upwards wins over the gravity force downwards, and the body will move upwards. In this case, the terminal velocity $u_c > 0$. When $\varrho_b > \varrho$, we get a motion downwards, and $u_c < 0$. The corresponding u is then also negative, but the scaled velocity u/u_c, becomes positive.

Inserting $u = u_c\bar{u} = b\bar{u}/a$ and $t = t_c\bar{t}$ in (9) leads to

$$\frac{d\bar{u}}{d\bar{t}} = -t_c a\bar{u} + \frac{t_c}{u_c}b, \quad \bar{u}(0) = I\frac{a}{b}.$$

We want the scales such that $d\bar{u}/d\bar{t}$ and $\bar{u}$ are about unity. To balance the size of $\bar{u}$ and $d\bar{u}/d\bar{t}$ we must therefore choose $t_c = 1/a$, resulting in the scaled ODE problem

$$\frac{d\bar{u}}{d\bar{t}} = -\bar{u} + 1, \quad u(0) = \beta, \tag{12}$$

where β is a dimensionless number,

$$\beta = \frac{I}{u_c} = I\frac{a}{b}, \tag{13}$$

reflecting the ratio of the initial velocity and the terminal ($t \to \infty$) velocity b/a. Scaling normally ends up with one or more dimensionless parameters, such as β here, containing ratios of physical effects in the model. Many more examples on dimensionless parameters will appear in later sections.

The analytical solution of the scaled model (12) reads

$$\bar{u_{\mathrm{e}}}(t) = e^{-t}\left(e^{t} - 1 + \beta\right) = 1 + (\beta - 1)e^{-t}\,. \tag{14}$$

The result (12) with the solution (14) is actually astonishing if a and b are as in (10): the six parameters d, μ, ϱ_b, ϱ, V, and g are conjured to one:

$$\beta = I\frac{3\pi d\mu}{\varrho_b V}\frac{1}{g}\left(\frac{\varrho}{\varrho_b} - 1\right)^{-1},$$

which is an enormous simplification of the problem if our aim is to investigate how u varies with the physical input parameters in the model. In particular, if the motion starts from rest, $\beta = 0$, and there are no physical parameters in the scaled model! We can then perform a single simulation and recover all physical cases by the unscaling procedure. More precisely, having computed $\bar{u}(\bar{t})$ from (12), we can use

$$u(t) = \frac{b}{a}\bar{u}(at), \tag{15}$$

to scale us back to the original problem again. We observe that (12) can utilize a solver for (9) by setting $a = 1$, $b = 1$, and $I = \beta$. Given some implementation of a solver for (9), say `solver(I, a, b, T, dt, theta)`, the scaled model is run by `solver(beta, 1, 1, T, dt, theta)`.

Software. We may develop a solver for the scaled problem that uses `joblib` to cache solutions with the same β, Δt, and T. For now we fix $\theta = 0.5$. The module `decay_vc.py`[10] has a function `solver(I, a, b, T, dt, theta)` for solving $u'(t) = -a(t)u(t) + b(t)$ for $t \in (0, T]$, $u(0) = I$, with time step `dt`. We reuse this function and call it with $a = b = 1$ and $I = \beta$ to solve the scaled problem:

```
from decay_vc import solver as solver_unscaled

def solver_scaled(beta, T, dt, theta=0.5):
    """
    Solve u'=-u+1, u(0)=beta for (0,T]
    with step dt and theta method.
    """
    print 'Computing the numerical solution'
    return solver_unscaled(
        I=beta, a=lambda t: 1, b=lambda t: 1,
        T=T, dt=dt, theta=theta)

import joblib
disk_memory = joblib.Memory(cachedir='temp')
solver_scaled = disk_memory.cache(solver_scaled)
```

If we want to plot the physical solution, we need an `unscale` function,

[10] `http://tinyurl.com/nm5587k/decay/decay_vc.py`

```
def unscale(u_scaled, t_scaled, d, mu, rho, rho_b, V):
    a, b = ab(d, mu, rho, rho_b, V)
    return (b/a)*u_scaled, a*t_scaled

def ab(d, mu, rho, rho_b, V):
    g = 9.81
    a = 3*pi*d*mu/(rho_b*V)
    b = g*(rho/rho_b - 1)
    return a, b
```

Looking at droplets of water in air, we can fix some of the parameters and let the size parameter d be the one for experimentation. The following function sets physical parameters, computes β, runs the solver for the scaled problem (`joblib` detects if it is necessary), and finally plots the scaled curve $\bar{u}(\bar{t})$ and the unscaled curve $u(t)$.

```
def main(dt=0.075, # Time step, scaled problem
         T=7.5,    # Final time, scaled problem
         d=0.001,  # Diameter (unscaled problem)
         I=0,      # Initial velocity (unscaled problem)
         ):
    # Set parameters, solve and plot
    rho = 0.00129E+3  # air
    rho_b = 1E+3      # density of water
    mu = 0.001        # viscosity of water
    # Asumme we have list or similar for d
    if not isinstance(d, (list,tuple,np.ndarray)):
        d = [d]

    legends1 = []
    legends2 = []
    plt.figure(1)
    plt.figure(2)
    betas = []     # beta values already computed (for plot)

    for d_ in d:
        V = 4*pi/3*(d_/2.)**3  # volume
        a, b = ab(d_, mu, rho, rho_b, V)
        beta = I*a/b
        # Restrict to 3 digits in beta
        beta = abs(round(beta, 3))

        print 'beta=%.3f' % beta
        u_scaled, t_scaled = solver_scaled(beta, T, dt)

        # Avoid plotting curves with the same beta value
        if not beta in betas:
            plt.figure(1)
            plt.plot(t_scaled, u_scaled)
            plt.hold('on')
            legends1.append('beta=%g' % beta)
        betas.append(beta)

        plt.figure(2)
        u, t = unscale(u_scaled, t_scaled, d_, mu, rho, rho_b, V)
        plt.plot(t, u)
        plt.hold('on')
        legends2.append('d=%g [mm]' % (d_*1000))
    plt.figure(1)
    plt.xlabel('scaled time'); plt.ylabel('scaled velocity')
    plt.legend(legends1, loc='lower right')
```

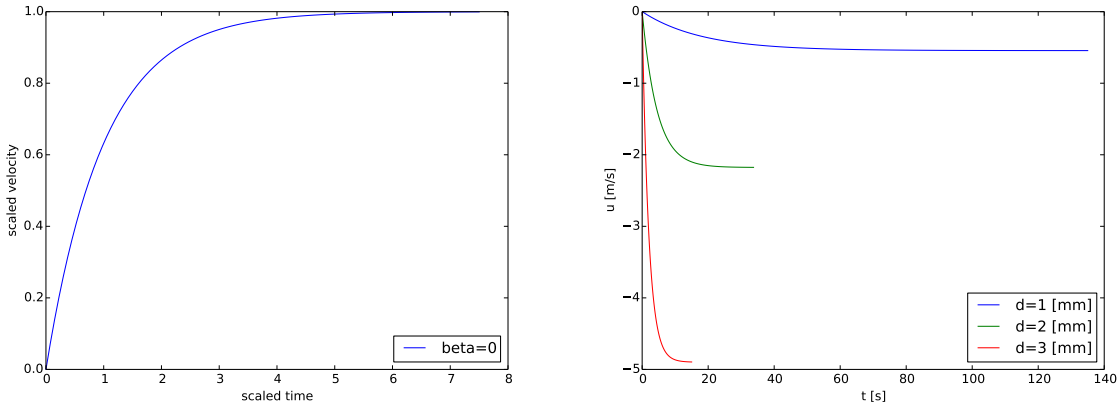

Figure 3: Velocity of falling body: scaled (left) and with dimensions (right).

The most complicated part of the code is related to plotting, but this part can be skipped when trying to understand how we work with a scaled model to perform the computations. The complete program is found in the file `falling_body.py`[11].

Since $I = 0$ implies $\beta = 0$, we can run different d values without any need to recompute $\bar{u}(\bar{t})$ as long as we assume the particle starts from rest.

From the scaling, we see that $u_c = b/a \sim d^{-2}$ and also that $t_c = 1/a \sim d^{-2}$, so plotting of $u(t)$ with dimensions for various d values will involve significant variations in the time and velocity scales. Figure 3 has an example with $d = 1, 2, 3$ mm, where we clearly see the different time and velocity scales in the figure with unscaled variables. Note that the scaled velocity is positive because of the sign of u_c (see the box above).

2.4 Variable coefficients

When a prescribed coefficient like $a(t)$ in $u'(t) = -a(t)u(t)$ varies with time one usually also performs a scaling of this a,

$$\bar{a}(\bar{t}) = \frac{a(t) - a_0}{a_c},$$

where the goal is to have the scaled $|\bar{a}|$ of size unity: $|\bar{a}| \leq 1$. This property is obtained by choosing a_c as the maximum value of $|a(t) - a_0|$ for $t \in [0, T]$, which is usually a quantity that can be estimated since $a(t)$ is known as a function of t. The a_0 parameter can be chosen as 0 here. (It could be tempting to choose $a_0 = \min_t a(t)$ so that $0 \leq \bar{a} \leq 1$, but then there is at least one point where $\bar{a} = 0$ and the differential equation collapses to $u' = 0$.)

As an example, imagine a decaying cell culture where we at time t_1 change the environment such that the death rate increases: $a(t) = d$ for $t < t_1$ and $a(t) = 5d$ for $t \geq t_1$. The model reads $u' = -a(t)u$, $u(0) = I$.

The $a(t)$ function is scaled by letting the characteristic size be $a_c = d$ (and $a_0 = 0$):

$$\bar{a}(\bar{t}) = \begin{cases} 1, & \bar{t} < t_1/t_c \\ 5, & \bar{t} \geq t_1/t_c \end{cases}$$

The scaled equation becomes

[11] `http://tinyurl.com/nm5587k/scale/falling_body.py`

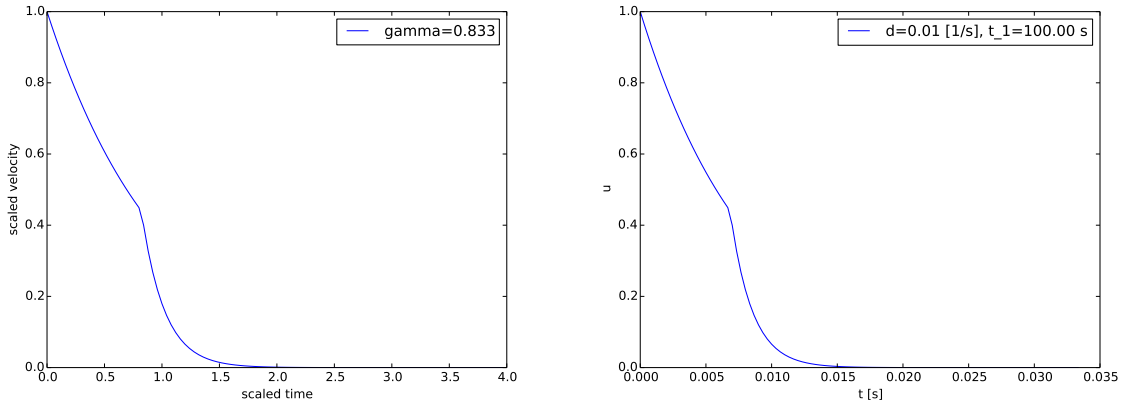

Figure 4: Exponential decay with jump.

$$\frac{u_c}{t_c}\frac{d\bar{u}}{d\bar{t}} = a_c\bar{a}(\bar{t})u_c\bar{u}, \quad u_c\bar{u}(0) = I\,.$$

The characteristic time, previously taken as $t_c = 1/a$, can now be taken as $t_c = t_1$ or $t_c = 1/d$. The natural choice of u_c is I. With $t_c = 1/d$ we get

$$\bar{u}'(\bar{t}) = -\bar{a}\bar{u}, \quad \bar{u}(0) = 1, \quad \bar{a} = \begin{cases} 1, & \bar{t} < \gamma \\ 5, & \bar{t} \geq \gamma \end{cases} \tag{16}$$

where

$$\gamma = t_1 d$$

is a dimensionless number in the problem. With $t_c = t_1$, we get

$$\bar{u}'(\bar{t}) = -\gamma\bar{a}\bar{u}, \quad \bar{u}(0) = 1, \quad \bar{a} = \begin{cases} 1, & \bar{t} < 1 \\ 5, & \bar{t} \geq 1 \end{cases}$$

The dimensionless parameter γ is now in the equation rather than in the definition of $\bar{a}$. Both problems involve γ, which is the ratio between the time when the environmental change happens and the typical time for the decay ($1/d$).

A computation with the scaled model (16) and the original model with dimensions appears in Figure 4.

2.5 Scaling a cooling problem with constant surroundings

The heat exchange between a body at temperature $T(t)$ and the surroundings at $T_s(t)$ can be modeled by Newton's law of cooling:

$$T'(t) = -k(T - T_s(t)), \quad T(0) = T_0, \tag{17}$$

where k is a prescribed heat transfer coefficient. An analytical solution is always handy to have as a control of the choice of scales. Here we have the result $T(t) = T_s + (T_0 - T_s)e^{-kt}$ when T_s is constant, which is also the assumption for now.

Physically, we expect the temperature to start at T_0 and then to move toward the surroundings (T_s). We therefore expect that T lies between T_0 and T_s. This is mathematically demonstrated by the analytical solution as well. A proper scaling is therefore to scale and translate T according to

$$\bar{T} = \frac{T - T_0}{T_s - T_0}\,. \tag{18}$$

Now, $0 \leq \bar{T} \leq 1$.

Scaling time by $\bar{t} = t/t_c$ and inserting $T = T_0 + (T_s - T_0)\bar{T}$ and $t = t_c\bar{t}$ in the problem (17) gives

$$\frac{d\bar{T}}{d\bar{t}} = -t_c k(\bar{T} - 1), \quad \bar{T}(0) = 0\,.$$

A natural choice, as argued in other exponential decay problems, is to choose $t_c k = 1$, which leaves us with the scaled problem

$$\frac{d\bar{T}}{d\bar{t}} = -(\bar{T} - 1), \quad \bar{T}(0) = 0\,. \tag{19}$$

No physical parameter enters this problem! Our scaling implies that $\bar{T}$ starts at 0 and approaches 1 as $\bar{t} \to \infty$, also in the case $T_s < T_0$. The physical temperature is always recovered as

$$T(t) = T_0 + (T_s - T_0)\bar{T}(k\bar{t})\,. \tag{20}$$

An implementation for (17) works for (19) by setting $k = 1$, $T_s = 1$, and $T_0 = 0$.

An alternative scaling is to choose

$$\bar{T} = \frac{T - T_s}{T_0 - T_s}\,. \tag{21}$$

Now $\bar{T} = 1$ initially and approaches zero as $t \to \infty$. The resulting scaled ODE problem then becomes

$$\frac{d\bar{T}}{d\bar{t}} = -\bar{T}, \quad \bar{T}(0) = 1\,. \tag{22}$$

2.6 Scaling a cooling problem with time-dependent surroundings

Let us apply the model (17) in case the surrounding temperature varies in time. Say we have an oscillating temperature environment according to

$$T_s(t) = T_m + a\sin(\omega t)\,. \tag{23}$$

Exact solution. It is possible to solve the differential equation problem analytically, and such a solution is a good help to see what scales are. In general, using the method of integrating factors for the original differential equation, we have

$$T(t) = T_0 e^{-kt} + e^{-kt}k\int_0^t e^{k\tau}T_s(\tau)d\tau\,.$$

With $T_s(t) = T_m + a\sin(wt)$ we can use SymPy to help us with integrations:

```
>>> from sympy import *
>>> t, k, T_m, a, w = symbols('t k T_m a w', real=True, positive=True)
>>> T_s = T_m + a*sin(w*t)
>>> I = exp(k*t)*T_s
>>> I = integrate(I, (t, 0, t))
>>> Q = k*exp(-k*t)*I
>>> Q = simplify(expand(Q))
>>> print Q
(-T_m*k**2 - T_m*w**2 + a*k*w +
(T_m*k**2 + T_m*w**2 + a*k**2*sin(t*w) -
a*k*w*cos(t*w))*exp(k*t))*exp(-k*t)/((k**2 + w**2))
```

Reordering the result, we get

$$T(t) = T_0 e^{-kt} + T_m(1 - e^{-kt}) + (k^2 + w^2)^{-1}(akwe^{-kt} + ak\sin(wt) - akw\cos(wt))\,.$$

Scaling. The scaling (18) brings in a time-dependent characteristic temperature scale $T_s - T_0$. Let us start with a fixed scale, where we take the characteristic temperature variation to be $T_m - T_0$:

$$\bar{T} = \frac{T - T_0}{T_m - T_0}\,.$$

We see from the analytical solution, and realize also by physical reasoning, that T sets out at T_0, but with time, it will oscillate around T_m. The typical average temperature span is therefore $|T_m - T_0|$.

We get from the differential equation, with $t_c = 1/k$ as in the former case,

$$k(T_m - T_0)\frac{d\bar{T}}{d\bar{t}} = -k((T_m - T_0)\bar{T} + T_0 - T_m - a\sin(wt),$$

resulting in

$$\frac{d\bar{T}}{d\bar{t}} = -\bar{T} + 1 + \alpha\sin(\beta\bar{t}), \quad \bar{T}(0) = 0, \tag{24}$$

where we have two dimensionless numbers:

$$\alpha = \frac{a}{T_m - T_0}, \quad \beta = \frac{w}{k}\,.$$

The α quantity measures the ratio of temperatures: amplitude of oscillations versus characteristic total temperature variation. The β number is the ratio of the two time scales: the frequency of the oscillations in T_s and the inverse e-folding time of the heat transfer. For clear interpretation of β we may introduce the period $P = 2\pi/w$ of the oscillations in T_s and the e-folding time $e = 1/k$. Then $\beta = 2\pi e/P$ and measures the period versus the e-folding time.

The original problem features five physical parameters: k, T_0, T_m, a, and w, but only two dimensionless numbers appear in the scaled model.

Software. Implementing the unscaled problem (17) can be reused for the scaled model by setting $k = 1$, $T_0 = 0$, and $T_s(t) = 1 + \alpha\sin(\beta\bar{t})$ ($T_m = 1$, $a = \alpha$, $w = \beta$).

Discussion of the time scale. Looking at the analytical insight we have, $T(t)$ has two characteristic terms in time: e^{-kt} and $\sin(wt)$. The former points to a time scale $t_c = 1/k$, while the latter to $t_c = 1/w$. Which one should be chosen? Bringing the temperature from T_0 to the level of the surroundings, T_m, goes like e^{-kt}, so in this process $t_c = 1/k$ is the characteristic time. Thereafter, the body's temperature just responds to the oscillations and the $\sin(wt)$ (and $\cos(wt)$) term dominates. For these large times, $t_c = 1/w$ is the appropriate time scale. Choosing $t_c = 1/w$ results in

$$\frac{d\bar{T}}{d\bar{t}} = -\beta^{-1}(\bar{T} - (1 + \alpha\sin(\bar{t}))), \quad \bar{T}(0) = 0\,. \tag{25}$$

Let us illustrate another, less effective, scaling. The temperature scale in (18) looks natural, so we apply this choice of scale. The characteristic temperature $T_0 - T_s$ now involves a time-dependent term $T_s(t)$. The mathematical steps become a bit more technically involved:

$$T(t) = T_0 + (T_s(t) - T_0)\bar{T},$$

$$\frac{dT}{dt} = \frac{dT_s}{dt}\bar{T} + (T_s - T_0)\frac{d\bar{T}}{d\bar{t}}\frac{d\bar{t}}{dt}\,.$$

With $\bar{t} = t/t_c = kt$ we get from the differential equation

$$\frac{dT_s}{dt}\bar{T} + (T_s - T_0)\frac{d\bar{T}}{dt}k = -k(\bar{T} - 1)(T_s - T_0),$$

which after dividing by $k(T_s - T_0)$ results in

$$\frac{d\bar{T}}{d\bar{t}} = -(\bar{T} - 1) - \frac{dT_s}{dt}\frac{\bar{T}}{k(T_s - T_0}),$$

or

$$\frac{d\bar{T}}{d\bar{t}} = -(\bar{T} - 1) - \frac{a\omega\cos(\omega\bar{t}/k)}{k(T_m + a\sin(\omega\bar{t}/k) - T_0)}\bar{T}\,.$$

The last term is complicated and becomes more tractable if we factor out dimensionless numbers. To this end, we scale T_s by (e.g.) T_m, which means to factor out T_m in the denominator. We are then left with

$$\frac{d\bar{T}}{d\bar{t}} = -(\bar{T} - 1) - \alpha\beta\frac{\cos(\beta\bar{t})}{1 + \alpha\sin(\beta\bar{t}) - \gamma}\bar{T}, \tag{26}$$

where α, β, and γ are dimensionless numbers characterizing the relative importance of parameters in the problem:

$$\alpha = a/T_m, \quad \beta = \omega/k, \quad \gamma = T_0/T_m\,. \tag{27}$$

We notice that (26) is not a special case of the original problem (17). Furthermore, the original five parameters k, T_m, a, ω, and T_0 are reduced to three dimensionless parameters. We conclude that this scaling is inferior, because using the temperature scale $T_0 - T_m$ enables reuse of the software for the unscaled problem and only two dimensionless parameters appear in the scaled model.

2.7 Scaling a nonlinear ODE

Exponential growth models, $u' = au$, are not realistic in environments with limited resources. The idea is then to assume that the growth rate a decreases with u and vanishes when we reach the maximum value M of u the environment can sustain. The initial growth rate is set to r: $a(0) = \varrho$. In general, this reasoning gives rise to models

$$u' = a(u)u, \quad u(0) = I, \tag{28}$$

with the logistic model, corresponding to $a(u) = \varrho(1 - u/M)$, as the simplest:

$$u' = \varrho u(1 - u/M), \quad u(0) = I\,. \tag{29}$$

A general choice of a might be $a(u) = \varrho(1 - u/M)^p$ for some exponent p.

Let us scale (28) with $a(u) = \varrho(1 - u/M)^p$. The natural scale for u is M ($u_c = M$), since we know that $0 < u \leq M$, and this makes the dimensionless $\bar{u} = u/M \in (0, 1]$. The function $a(u)$ is typically varying between 0 and ϱ, so it can be scaled as

$$\bar{a}(\bar{u}) = \frac{a(u)}{\varrho} = (1 - \frac{u}{M})^p = (1 - \bar{u})^p\,.$$

Time is scaled as $\bar{t} = t/t_c$ for some suitable characteristic time t_c. Inserted in (28), we get

$$\frac{u_c}{t_c}\frac{d\bar{u}}{d\bar{t}} = \varrho\bar{a}u_c\bar{u}, \quad u_c\bar{u}(0) = I,$$

resulting in

$$\frac{d\bar{u}}{d\bar{t}} = t_c\varrho(1 - \bar{u})^p\bar{u}, \quad \bar{u}(0) = \frac{I}{M}\,.$$

A natural choice is $t_c = 1/\varrho$ as in other exponential growth models since it leads to the term on the right-hand side to be about unity, as the left-hand side, if the scaling is physically correct. Introducing the dimensionless parameter

$$\alpha = \frac{I}{M},$$

measuring the fraction of the initial population compared to the maximum one, we get the dimensionless model

$$\frac{d\bar{u}}{d\bar{t}} = (1 - \bar{u})^p\bar{u}, \quad \bar{u}(0) = \alpha\,. \tag{30}$$

Here, we have two dimensionless parameters: α and p. A classical logistic model with $p = 1$ has only one dimensionless variable.

We could try another scaling of u where we also translate $\bar{u}$:

$$\bar{u} = \frac{u - I}{M}\,.$$

This choice of $\bar{u}$ results in

$$\frac{d\bar{u}}{d\bar{t}} = (1 - \alpha - \bar{u})^p\bar{u}, \quad \bar{u}(0) = 0\,. \tag{31}$$

The essential difference between (30) and (31) is that $\bar{u} \in [\alpha, 1]$ in the former and $\bar{u} \in [0, 1 - \alpha]$ in the latter. Both models involve the dimensionless numbers α and p. An advantage of (30) is

that software for the unscaled model can easily be used for the scaled model by choosing $I = \alpha$, $M = 1$, and $\varrho = 1$.

2.8 ODE systems for spreading of diseases

The field of epidemiology frequently applies ODE systems to describe the spreading of diseases, such as smallpox, measles, plague, ordinary flu, swine flu, and HIV. Different models include different effects, which are reflected in dimensionless numbers. Most of the effects are modeled as exponential decay or growth of the dependent variables.

SIR model. The model has three categories of people: susceptibles (S) who can get the disease, infected (I) who are infected and may infect susceptibles, and recovered (R) who have recovered from the disease and gained immunity. We introduce $S(t)$, $I(t)$, and $R(t)$ as the number of people in the categories S, I, and R, respectively. The model, naturally known as the SIR model, takes the form a system of ODEs:

$$\frac{dS}{dt} = -\beta SI, \tag{32}$$

$$\frac{dI}{dt} = \beta SI - \nu I, \tag{33}$$

$$\frac{dR}{dt} = \nu I, \tag{34}$$

where β and nu are empirical constants. The average time for recovering from the disease can be shown to be ν^{-1}, but β is much harder to estimate, so working with a scaled model where k is "scaled away" is advantageous. Adding (32)-(34) shows that

$$\frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt} = 0 \quad \Rightarrow \quad S + I + R = \text{const} = N,$$

where N is the size of the population. It is natural to scale S, I, and R by, e.g., $S(0)$:

$$\bar{S} = \frac{S}{S(0)}, \quad \bar{I} = \frac{I}{S(0)}, \quad \bar{R} = \frac{R}{S(0)}\,.$$

Introducing $\bar{t} = t/t_c$, we arrive at the equations

$$\frac{d\bar{S}}{d\bar{t}} = -t_c S(0)\beta\bar{S}\bar{I},$$

$$\frac{d\bar{I}}{d\bar{t}} = t_c S(0)\beta\bar{S}\bar{I} - t_c\nu\bar{I},$$

$$\frac{d\bar{R}}{d\bar{t}} = t_c\nu I,$$

with initial conditions $\bar{S}(0) = 1$, $\bar{I}(0) = I_0/S(0) = \alpha$, and $\bar{R}(0) = R(0)/S(0)$. Normally, $R(0) = 0$.

Taking $t_c = 1/\nu$, corresponding to a time unit equal to the time it takes to recover from the disease, we end up with the scaled model

$$\frac{d\bar{S}}{d\bar{t}} = -R_0\bar{S}\bar{I}, \tag{35}$$

$$\frac{d\bar{I}}{d\bar{t}} = R_0\bar{S}\bar{I} - \bar{I}, \tag{36}$$

$$\frac{d\bar{R}}{d\bar{t}} = I, \tag{37}$$

with $\bar{S}(0) = 1$, $\bar{I}(0) = \alpha$, $\bar{R}(0) = 0$, and R_0 as the dimensionless number

$$R_0 = \frac{S(0)\beta}{\nu}\,. \tag{38}$$

We see from (36) that to make the disease spreading, $d\bar{I}/d\bar{t} > 0$, and therefore $R_0S(0) - 1 > 0$ or $R_0 > 1$ since $S(0) = 1$. Therefore, R_0 reflects the disease's ability to spread and is consequently an important dimensionless quantity, known as the *basic reproductive number*. **hpl 1**: Explain interpretation.

Looking at (33), we see that to increase I initially, we must have $dI/dt > 0$ at $t = 0$, which implies $\beta I(0)S(0) - \nu I(0) > 0$, i.e., $R_0 > 1$.

We can also scale S, I, and R by the total population $N = S(0) + I(0) + R(0)$,

$$\bar{S} = \frac{S}{N}, \quad \bar{I} = \frac{I}{N}, \quad \bar{R} = \frac{R}{N)}\,.$$

With the same time scale, one gets the system (35)-(37), but with R_0 replaced by the dimensionless number:

$$\tilde{R}_0 = \frac{N\beta}{\nu}\,. \tag{39}$$

The initial conditions become $\bar{S}(0) = 1 - \alpha$, $\bar{I}(0) = \alpha$, and $\bar{R}(0) = 0$.

For the disease to spread at $t = 0$, we must have $\tilde{R}_0\bar{S}(0) > 1$, but $\tilde{R}_0\bar{S}(0) = N\beta/\nu \cdot S(0)/N = R_0$, so the criterion is still $R_0 > 1$. Since R_0 is a more famous number than $\tilde{R}_0$, we can write the ODEs with $R_0/S(0) = R_0/(1-\alpha)$ instead of $\tilde{R}_0$.

Choosing t_c to make the SI terms balance the time derivatives, $t_c = (N\beta)^{-1}$, moves $\tilde{R}_0$ (or R_0 if we scale S, I, and R by $S(0)$) to the I terms:

$$\frac{d\bar{S}}{d\bar{t}} = -\bar{S}\bar{I},$$

$$\frac{d\bar{I}}{d\bar{t}} = \bar{S}\bar{I} - \tilde{R}_0^{-1}\bar{I},$$

$$\frac{d\bar{R}}{d\bar{t}} = \tilde{R}_0^{-1}I\,.$$

SIRV model with finite immunity. A common extension of the SIR model involves finite immunity: after some period of time, recovered individuals lose their immunity and become susceptibles again. This is modeled as a leakage $-\mu R$ from the R to the S category, where μ^{-1} is the average time it takes to lose immunity. Vaccination is another extension: a fraction pS is removed from the S category by successful vaccination and brought to a new category V (the vaccinated). The ODE model reads

$$\frac{dS}{dt} = -\beta SI - pS + \mu R, \tag{40}$$
$$\frac{dI}{dt} = \beta SI - \nu I, \tag{41}$$
$$\frac{dR}{dt} = \nu I - \mu R, \tag{42}$$
$$\frac{dV}{dt} = pS\,. \tag{43}$$

Using $t_c = 1/\nu$ and scaling the unknowns by $S(0)$ leads to the dimensionless model

$$\frac{d\bar{S}}{d\bar{t}} = -R_0\bar{S}\bar{I} - \delta S + \gamma\bar{R}, \tag{44}$$
$$\frac{d\bar{I}}{d\bar{t}} = R_0\bar{S}\bar{I} - \bar{I}, \tag{45}$$
$$\frac{d\bar{R}}{d\bar{t}} = \bar{I} - \gamma\bar{R}, \tag{46}$$
$$\frac{d\bar{V}}{d\bar{t}} = \delta\bar{S}, \tag{47}$$

with two new dimensionless parameters:

$$\gamma = \frac{\mu}{\nu}, \quad \delta = \frac{p}{\nu}\,.$$

The quantity p^{-1} can be interpreted as the average time it takes to vaccinate a susceptible successfully. Writing $\gamma = \nu^{-1}/\mu^{-1}$ and $\delta = \nu^{-1}/p^{-1}$ gives the interpretation that γ is the ratio of the average time to recover and the average time to lose immunity, while δ is the ratio of the average time to recover and the average time to successfully vaccinate a susceptible.

2.9 Michaelis-Menten kinetics for biochemical reactions

A classical reaction model in biochemistry describes how a substrate S is turned into a product P with aid of an enzyme E. S and E react to form a complex ES in the first stage of the reaction. In the second stage, ES is turned into E and P. Introducing the amount of S, E, ES, and P by $[S]$, $[E]$, $[ES]$, and $[P]$, the mathematical model can be written as

$$\frac{d[ES]}{dt} = k_+[E][S] - k_v[ES] - k_-[ES], \tag{48}$$
$$\frac{d[P]}{dt} = k_v[ES], \tag{49}$$
$$\frac{d[S]}{dt} = -k_+[E][S] + k_-[ES], \tag{50}$$
$$\frac{d[E]}{dt} = -k_+[E][S] + k_-[ES] + k_v[ES]\,. \tag{51}$$

The initial conditions are $[ES](0) = [P](0) = 0$, and $[S] = S_0$, $[E] = E_0$. Three rate constants are involved: k_+, k_-, and k_v.

The amount of substance is measured in the unit mole[12] (mol). From the equations we can see that k_+ is measured in $\mathrm{s}^{-1}\mathrm{mol}^{-1}$, while k_- and k_v are measured in s^{-1}. It is convenient to get rid of the mole unit for the amount of a substance. When working with dimensionless quantities, only ratios of the rate constants and not their specific values are needed.

Classical analysis. The typical analysis of the present ODE system is to first observe two conservation equations, arising from simply adding the ODEs:

$$\frac{d[ES]}{dt} + \frac{d[E]}{dt} = 0, \tag{52}$$

$$\frac{d[ES]}{dt} + \frac{d[S]}{dt} + \frac{d[P]}{dt} = 0, \tag{53}$$

from which it follows that

$$[ES] + [E] = E_0, \tag{54}$$

$$[ES] + [S] + [P] = S_0\,. \tag{55}$$

Using (54), we can eliminate $[E]$ and obtain a system of only two ODEs,

$$\frac{d[ES]}{dt} = k_+([ES] - E_0)[S] - (k_v + k_-)[ES], \tag{56}$$

$$\frac{d[S]}{dt} = -k_+([ES] - E_0)[S] + k_-[ES]\,. \tag{57}$$

A common assumption is that the formation of $[ES]$ is very fast and that it reaches an equilibrium state, $[ES]' = 0$. This implies

$$k_+([ES] - E_0)[S] - (k_v + k_-)[ES] = 0 \quad \Rightarrow \quad [ES] = \frac{E_0[S]}{[S] - K},$$

where

$$K = \frac{k_- + k_v}{k_+},$$

is the Michaelis constant. Using the expression for $[ES]$ in the equation for $[S]$ gives

$$\frac{dS}{dt} = \frac{k_v E_0[S]}{[S] + K}\,. \tag{58}$$

We see that the parameter K is central.

Dimensionless ODE system. Let us reason how to make the original ODE system dimensionless. Aiming at $[S]$ and $[E]$ of unit size, two obvious dimensionless unknowns are

$$\bar{S} = \frac{[S]}{S_0}, \quad \bar{E} = \frac{[E]}{E_0}\,.$$

For the other two unknowns we just introduce scales to be determined later:

[12] `https://en.wikipedia.org/wiki/Mole_(unit)`

$$\bar{P} = \frac{[P]}{P_c}, \quad \bar{Q} = \frac{[ES]}{Q_c}\,.$$

With $\bar{t} = t/t_c$ the equations become

$$\begin{aligned}
\frac{d\bar{Q}}{d\bar{t}} &= t_c k_+ \frac{E_0 S_0}{Q_c}\bar{E}\bar{S} - t_c(k_v + k_-)\bar{Q},\\
\frac{d\bar{P}}{d\bar{t}} &= t_c k_v \frac{Q_c}{P_c}\bar{Q},\\
\frac{d\bar{S}}{d\bar{t}} &= -t_c k_+ E_0 \bar{E}\bar{S} + t_c k_- \frac{Q_c}{S_0}\bar{Q},\\
\frac{d\bar{E}}{d\bar{t}} &= -t_c k_+ S_0 \bar{E}\bar{S} + t_c(k_- + k_v)\frac{Q_c}{E_0}\bar{Q}\,.
\end{aligned}$$

Determining scales. Choosing the scales is actually a quite complicated matter that requires extensive analysis of the equations to determine the characteristics of the solutions. Much literature is written about this, but here we shall take a simplistic and pragmatic approach. Besides the Michaelis constant, there is another important parameter,

$$\epsilon = \frac{E_0}{S_0},$$

because most applications will involve a small ϵ. We shall have K and ϵ in mind while choosing scales such that these symbols appear naturally in the scaled equations.

Looking at the equations, we see that the K parameter will appear if $t_c \sim 1/k_+$. However, $1/k_+$ does not have the dimension $[\mathrm{T}]^{-1}$ as required, so we need to add a factor with dimension mol. A natural choice is $t_c^{-1} = k_+ S_0$ or $t_c^{-1} = k_+ E_0$. Since often $S_0 \gg E_0$, the former t_c is a short time scale and the latter is a long time scale. If the interest is in the long time scale, we set

$$t_c = \frac{1}{k_+ E_0}\,.$$

The equations then take the form

$$\begin{aligned}
\frac{d\bar{Q}}{d\bar{t}} &= \frac{S_0}{Q_c}\bar{E}\bar{S} - K E_0^{-1}\bar{Q},\\
\frac{d\bar{P}}{d\bar{t}} &= \frac{k_v}{k_+ E_0}\frac{Q_c}{P_c}\bar{Q},\\
\frac{d\bar{S}}{d\bar{t}} &= -\bar{E}\bar{S} + \frac{k_-}{k_+ E_0}\frac{Q_c}{S_0}\bar{Q},\\
\frac{d\bar{E}}{d\bar{t}} &= -\epsilon^{-1}\bar{E}\bar{S} + K\frac{Q_c}{E_0^2}\bar{Q}\,.
\end{aligned}$$

The $[ES]$ variable starts and ends at zero, and its maximum value can be roughly estimated from the equation for $[ES]'$ by setting $[ES]' = 0$, which gives an estimate of

$$Q_c = \frac{E_0 S_0}{K},$$

if we approximate $[E][S]$ by $E_0 S_0$.

The equation for $\bar{P}$ simplifies if we choose $P_c = Q_c$. With these assumptions one gets

$$\begin{aligned}
\frac{d\bar{Q}}{d\bar{t}} &= KE_0^{-1}(\bar{E}\bar{S} - \bar{Q}), \\
\frac{d\bar{P}}{d\bar{t}} &= \frac{k_v}{k_+ E_0}\bar{Q}, \\
\frac{d\bar{S}}{d\bar{t}} &= -\bar{E}\bar{S} + \frac{k_-}{k_+ E_0}\frac{E_0}{K}\bar{Q}, \\
\frac{d\bar{E}}{d\bar{t}} &= -\epsilon^{-1}\bar{E}\bar{S} + \epsilon^{-1}\bar{Q}\,.
\end{aligned}$$

We can now identify the dimensionless numbers

$$\alpha = \frac{K}{E_0}, \quad \beta = \frac{k_v}{k_+ E_0}, \quad \gamma = \frac{k_-}{k_+ E_0},$$

where we see that $\alpha = \beta + \gamma$, so γ can be eliminated, leading to the final set of equations:

$$\frac{d\bar{Q}}{d\bar{t}} = \alpha(\bar{E}\bar{S} - \bar{Q}), \tag{59}$$

$$\frac{d\bar{P}}{d\bar{t}} = \beta\bar{Q}, \tag{60}$$

$$\frac{d\bar{S}}{d\bar{t}} = -\bar{E}\bar{S} + (1 - \beta\alpha^{-1})\bar{Q}, \tag{61}$$

$$\epsilon\frac{d\bar{E}}{d\bar{t}} = -\bar{E}\bar{S} + \bar{Q}\,. \tag{62}$$

The five initial parameters (S_0, E_0, k_+, k_-, and k_v) are reduced to three dimensionless constants:

- α is the dimensionless Michaelis constant, reflecting the ratio of the production of P and E ($k_v + k_-$) versus the production of the complex (k_+), made dimensionless by E_0,
- ϵ is the initial fraction of enzyme relative to the substrate,
- β measures the relative importance of production of P (k_v) versus production of the complex (k_+), made dimensionless by E_0.

Observe that software developed for solving (48)-(51) cannot be reused for solving (59)-(62) since the latter system has a slightly different structure.

Analysis of the scaled system. In the scaled system, we may assume ϵ small, which from (62) gives rise to the simplification $\epsilon\bar{E}' = 0$, and thereby the relation $\bar{Q} = \bar{E}\bar{S}$. The conservation equation $[ES] + [E] = E_0$ reads $Q_c\bar{Q} + E_0\bar{E} = E_0$ such that $\bar{E} = 1 - Q_c\bar{Q}/E_0 = 1 - \bar{Q}S_0/K = 1 - \epsilon^{-1}\alpha^{-1}\bar{Q}$. The relation $\bar{Q} = \bar{E}\bar{S}$ then becomes

$$\bar{Q} = (1 - \epsilon^{-1}\alpha^{-1}\bar{Q})\bar{S},$$

which can be solved for $\bar{Q}$:

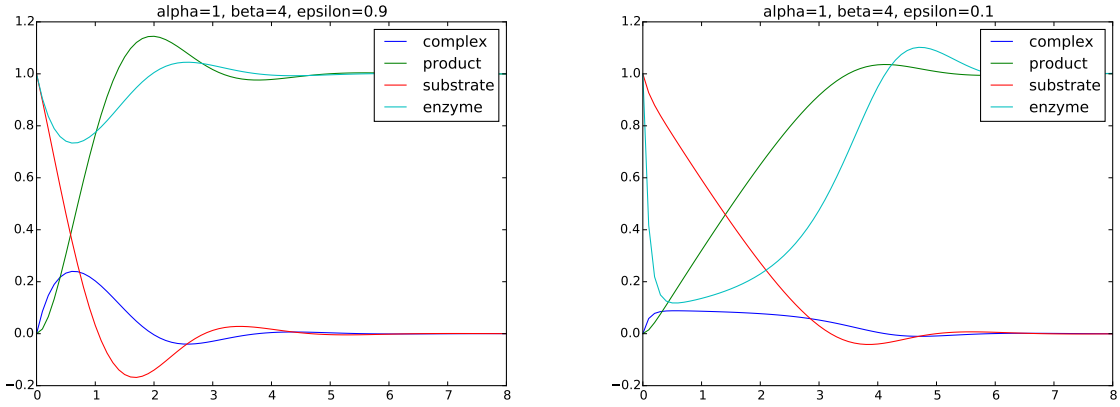

Figure 5: Simulation of a biochemical process.

$$\bar{Q} = \frac{\bar{S}}{1 + \epsilon^{-1}\alpha^{-1}\bar{S}}\,.$$

The equation (61) for $\bar{S}$ becomes

$$\frac{d\bar{S}}{d\bar{t}} = -\beta\alpha^{-1}\bar{Q} = -\frac{\beta\bar{S}}{\alpha + \epsilon^{-1}\bar{S}}\,. \tag{63}$$

This is a more precise analysis than the one leading to (58) since we now realize that the mathematical assumption for the simplification is $\epsilon \to 0$.

Is (63) consistent with (58)? It is easy to make algebraic mistakes when deriving scaled equations, so it is always wise to carry out such consistency checks. Introducing dimensions in (63) leads to

$$\frac{t_c}{S_0}\frac{dS}{dt} = \frac{d\bar{S}}{d\bar{t}} = -\frac{\beta\bar{S}}{\alpha + \epsilon^{-1}\bar{S}} = \frac{k_v}{k_+E_0}\frac{S}{KE_0^{-1} + E_0^{-1}S_0\bar{S}} = \frac{k_v}{k_+}\frac{\bar{S}}{K+S},$$

and hence with $t_c^{-1} = k_+E_0$,

$$\frac{dS}{dt} = \frac{k_vE_0S}{K+S},$$

which is (58).

Figure 5 shows the impact of ϵ: with a small value (0.1) we see that $\bar{Q} \approx 0$, which justifies the simplifications performed above. We also observe that all the unknowns vary between 0 and about 1, indicating that the scaling is successful for the chosen dimensionless numbers.

3 Vibration problems

We shall address a range of different second-order ODEs for mechanical vibrations and demonstrate how to reason about the scaling in different physical scenarios.

hpl 2: Include more figures.

3.1 Undamped vibrations without forcing

The simplest differential equation model for mechanical vibrations reads

$$mu'' + ku = 0, \quad u(0) = I,\ u'(0) = V\,. \tag{64}$$

This is a common model for a vibrating body with mass m attached to a linear spring with spring constant k (and force $-ku$). The quantity $u(t)$ measures the displacement of the body.

The first technical steps of scaling. The problem (64) has one independent variable t and one dependent variable u. We introduce dimensionless versions of these variables:

$$\bar{u} = \frac{u}{u_c}, \quad \bar{t} = \frac{t}{t_c},$$

where u_c and t_c are characteristic values of u and t. Inserted in (64), we get

$$m\frac{u_c}{t_c^2}\frac{d^2\bar{u}}{d\bar{t}^2} + ku_c\bar{u} = 0, \quad u_c\bar{u}(0) = I, \quad \frac{u_c}{t_c}\frac{d\bar{u}}{d\bar{t}}(0) = V,$$

resulting in

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \frac{t_c^2 k}{m}\bar{u} = 0, \quad \bar{u}(0) = \frac{I}{u_c},\ \bar{u}'(0) = \frac{Vt_c}{u_c}\,. \tag{65}$$

The exact solution. What is an appropriate displacement scale u_c? The initial condition $u(0) = I$ is a candidate, i.e., $u_c = I$. We also have analytical insight into this problem, because the exact solution of (64) is quite straightforward to compute. Let us demonstrate the steps in a SymPy session:

```
from sympy import *
u = symbols('u', cls=Function)
w = symbols('w', real=True, positive=True)
I, V, C1, C2 = symbols('I V C1 C2', real=True)
# Define differential equation: u'' + w**2*u = 0
def ode(u):
    return diff(u, t, t) + w**2*u

diffeq = ode(u(t))
# Solve differential equation with dsolve
s = dsolve(diffeq, u(t))
# s is an u(t) == expression (Eq obj.), s.rhs grabs the expression
u_sol = s.rhs
# The solution contains integration constants C1 and C2
# but these are not symbols, substitute them by symbols
u_sol = u_sol.subs('C1', C1).subs('C2', C2)
# Determine C1 and C2 from the initial conditions
ic = [u_sol.subs(t, 0) - I, u_sol.diff(t).subs(t, 0) - V]
print ic
s = solve(ic, [C1, C2])
# s is now a dictionary, substitute solution back in u_sol
u_sol = u_sol.subs(C1, s[C1]).subs(C2, s[C2])
print u_sol

# Check that the solution fulfills the ODE and init.cond.
print simplify(ode(u_sol)),
print u_sol.subs(t, 0) - I, diff(u_sol, t).subs(t, 0) - V
```

The output is `I*cos(t*w) + V*sin(t*w)/w`, so we have found that

$$u(t) = I\cos(\omega t) + \frac{V}{\omega}\sin(\omega t), \quad \omega = \sqrt{\frac{k}{m}}\,.$$

More insight arises from rewriting such an expression in the form $A\cos(wt-\phi)$:

$$u(t) = \sqrt{I^2 + \frac{V^2}{\omega^2}}\cos(wt-\phi), \quad \phi = \tan^{-1}(V/(\omega I))\,.$$

Now we see that the u corresponds to cosine oscillations with a phase shift ϕ and amplitude $\sqrt{I^2+(V/\omega)^2}$.

Discussion of the displacement scale. The amplitude of u is $\sqrt{I^2+V^2/\omega^2}$, and this expression is obviously a candidate for u_c. However, the simpler choice $u_c = \max(I, V/\omega)$ is also relevant and more attractive than the square root expression (but potentially a factor 1.4 wrong compared to the exact amplitude). It is not very important to have $|u| \leq 1$, the point is to avoid $|u|$ very small or large.

Discussion of the time scale. What is an appropriate time scale? Looking at (65) and arguing that $\bar{u}''$ and $\bar{u}$ both should be around unity in size, the coefficient $t_c^2 k/m$ must equal unity, implying that $t_c = \sqrt{m/k}$. Also from the analytical solution we see that it goes like sine or cosine of ωt so $1/\omega = \sqrt{m/k}$ can be a characteristic time scale. Likewise, one period of the oscillations, $P = 2\pi/\omega$ can be the characteristic time, leading to $t_c = 2\pi/\omega$.

With $u_c = I$ and $t_c = \sqrt{m/k}$ we get the scaled model

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \bar{u} = 0, \quad \bar{u}(0) = 1,\ \bar{u}'(0) = \alpha, \tag{66}$$

where α is a dimensionless parameter:

$$\alpha = \frac{V}{I}\sqrt{\frac{m}{k}}\,.$$

Note that in case $V = 0$, we have "scaled away" all physical parameters. The universal solution without physical parameters is then $\bar{u}(\bar{t}) = \cos\bar{t}$.

The unscaled solution is recovered as

$$u(t) = I\bar{u}(\sqrt{k/m}\bar{t})\,. \tag{67}$$

This expressions shows that the scaling is simply a matter of *stretching or shrinking the axes.*

Alternative displacement scale. Using $u_c = V/\omega$, the equation is not changed, but the initial conditions become

$$\bar{u}(0) = \frac{I}{u_c} = \frac{I\omega}{V} = \frac{I}{V}\sqrt{\frac{k}{m}} = \alpha^{-1}, \quad \bar{u}'(0) = 1\,.$$

With $u_c = V/\omega$ and one period as time scale, $t_c = 2\pi\sqrt{m/k}$, we get the alternative model

$$\frac{d^2\bar{u}}{d\bar{t}^2} + 4\pi^2\bar{u} = 0, \quad \bar{u}(0) = \alpha^{-1},\ \bar{u}'(0) = 2\pi\,. \tag{68}$$

The unscaled solution is in this case recovered by

$$u(t) = V\sqrt{\frac{m}{k}}\bar{u}(2\pi\sqrt{k/m}\bar{t})\,. \tag{69}$$

About frequency and dimensions. The solution goes like $\cos\omega t$, where $\omega = \sqrt{m/k}$ must have dimension 1/s. Actually, ω has dimension *radians per second*: rad/s. A radian is dimensionless since it is arc (length) divided by radius (length), but still regarded as a unit. The period P of vibrations is a more intuitive quantity than the frequency ω. The relation between P and ω is $P = 2\pi/\omega$. The number of oscillation cycles per period, f, is a more intuitive measurement of frequency and also known as *frequency*. Therefore, to be precise, ω should be named *angular frequency*. The relation between f and T is $f = 1/T$, so $f = 2\pi\omega$ and measured in Hz (1/s), which is the unit for counts per unit time.

3.2 Undamped vibrations with constant forcing

For vertical vibrations in the gravity field, the model (64) must also take the gravity force $-mg$ into account: $mu'' + ku = -mg$. How does this term influence the scaling? We observe that if there is no movement of the body, $u'' = 0$, and the spring elongation matches the gravity force: $ku = -mg$, leading to a steady displacement $u = -mg/k$. We can then have oscillations around this equilibrium point. A natural scaling for u is therefore

$$\bar{u} = \frac{u - (-mg/k)}{u_c} = \frac{uk + mg}{ku_c}\,.$$

The scaled differential equation with the same time scale as before (gravity does not influence the oscillations) reads

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \bar{u} - \frac{t_c^2}{u_c}g = -\frac{t_c^2}{u_c}g,$$

leading to

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \bar{u} = 0\,.$$

3.3 Undamped vibrations with time-dependent forcing

Now we add a transient forcing term $F(t)$ to the model (64):

$$mu'' + ku = F(t), \quad u(0) = I,\ u'(0) = V\,. \tag{70}$$

Take the forcing to be oscillating

$$F(t) = A\cos(\psi t)\,.$$

The technical steps of the scaling are still the same,

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \frac{t_c^2 k}{m}\bar{u} = \frac{t_c^2}{mu_c}A\cos(\psi t_c\bar{t}), \quad \bar{u}(0) = \frac{I}{u_c},\ \bar{u}'(0) = \frac{Vt_c}{u_c}\,. \tag{71}$$

What are typical displacement and time scales? This is not so obvious without knowing the details of the solution, because there are three parameters (I, V, and A) that influence the magnitude of u. Moreover, there are two time scales, one for the free vibrations of the systems and one for the forced vibrations $F(t)$.

Investigating scales via analytical solutions. We may look into the exact solution to see what the scales are. We continue the SymPy session from the previous section and perform much of the same steps. Note that we use `w` for $\omega = \sqrt{k/m}$. SymPy may get confused when coefficients in differential equations contain several symbols. We therefore rewrite the equation with at most one symbol in each coefficient. The amplitude A/m in the forcing term is of this reason replaced by the symbol `A1`.

```
A, A1, m, psi = symbols('A A1 m psi', positive=True, real=True)
def ode(u):
    return diff(u, t, t) + w**2*u - A1*cos(psi*t)

diffeq = ode(u(t))
u_sol = dsolve(diffeq, u(t))
u_sol = u_sol.rhs

# Determine the constants C1 and C2 in u_sol
# (first substitute our own declared C1 and C2 symbols,
# then use the initial conditions)
u_sol = u_sol.subs('C1', C1).subs('C2', C2)
eqs = [u_sol.subs(t, 0) - I, u_sol.diff(t).subs(t, 0) - V]
s = solve(eqs, [C1, C2])
u_sol = u_sol.subs(C1, s[C1]).subs(C2, s[C2])

# Check that the solution fulfills the equation and init.cond.
print simplify(ode(u_sol))
print simplify(u_sol.subs(t, 0) - I)
print simplify(diff(u_sol, t).subs(t, 0) - V)

u_sol = simplify(expand(u_sol.subs(A1, A/m)))
print u_sol
```

The output from the last line is

```
A/m*cos(psi*t)/(-psi**2 + w**2) + V*sin(t*w)/w +
(A/m + I*psi**2 - I*w**2)*cos(t*w)/(psi**2 - w**2)
```

With a bit of rewrite this expressions means

$$u(t) = \frac{A/m}{\omega^2 - \psi^2}\cos(\psi t) + \frac{V}{\omega}\sin(\omega t) + \left(\frac{A/m}{\psi^2 - \omega^2} + I\right)\cos(\omega t)\,.$$

Obviously, this expression is only meaningful for $\psi \neq \omega$. The case $\psi = \omega$ gives an infinite amplitude in this model, a phenomenon known as resonance. The amplitude becomes finite when damping is included, see Section 3.4.

For the case the system starts from rest, $I = V = 0$, and the forcing is the only driving mechanism, we can simplify:

$$\begin{aligned} u(t) &= \frac{A}{m(\omega^2 - \psi^2)}\cos(\psi t) + \frac{A}{m(\psi^2 - \omega^2)}\cos(\omega t) \\ &= \frac{A}{m(\omega^2 - \psi^2)}(\cos(\psi t) - \cos(\omega t))\,. \end{aligned}$$

To gain more insight, $\cos(\psi t) - \cos(\omega t)$ can be rewritten in terms of the mean frequency $(\psi + \omega)/2$ and the difference in frequency $(\psi - \omega)/2$:

$$u(t) = \frac{A}{m(\omega^2 - \psi^2)} 2\sin\left(\frac{\psi - \omega}{2}t\right)\sin\left(\frac{\psi + \omega}{2}t\right), \tag{72}$$

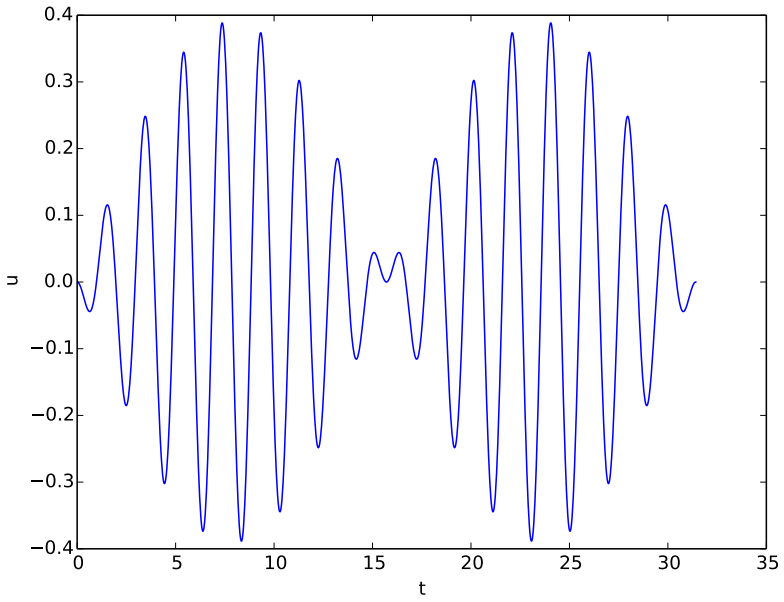

Figure 6: Signal with frequency 3.1 and envelope frequency 0.2.

showing that there is a signal with frequency $(\psi + \omega)/2$ whose amplitude has a (much) slower frequency $(\psi - \omega)/2$. Figure 6 shows an example how such a signal might look like.

The displacement and time scales. A characteristic displacement can in the latter special case be taken as $u_c = A/(m(\omega^2 - \psi^2))$. This is also a relevant choice in the more general case $I \neq 0, V \neq 0$, unless I or V is so large that it dominates the amplitude caused by the forcing. With $u_c = A/(m(\omega^2 - \psi^2))$ we also have three special cases: $\omega \ll \psi$, $\omega \gg \psi$, and $\psi \sim \omega$. In the latter case we need $u_c = A/(m(\omega^2 - \psi^2))$ if we want $|u| \leq 1$. When ω and ψ are significantly different, we may choose one of them, say ω, and $u_c = A/k$. This is also the relevant scale if $\omega \gg \psi$. In the opposite case, $\omega \ll \psi$, $u_c = A/(m\psi^2)$.

The time scale is dominated by the fastest oscillations, which are of frequency ψ or ω when these are close and the largest of them when they are distant. In any case, we might choose $t_c = 1/\max(\psi, \omega)$.

Finding the displacement scale from the differential equation. Going back to (71), we may demand that all the three terms in the differential equation are of size unity. This leads to $t_c = \sqrt{m/k}$ and $u_c = At_c^2/m = A/k$. The formula for u_c is a kind of measure of the ratio of the forcing and the spring force (the dimensionless number $A/(ku_c)$ would be this ratio).

Looking at (72), we see that if $\psi \ll \omega$, the amplitude can be approximated by $A/(m\omega^2) = A/k$, showing that the scale $u_c = A/k$ is relevant for small excitation frequency ψ compared to the free vibration frequency ω.

Scaling with free vibrations as time scale. We first choose the time scale based on the free oscillations with frequency ω, i.e., $t_c = 1/\omega$. Inserting the expression in (71) results in

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \bar{u} = \gamma\cos(\delta\bar{t}), \quad \bar{u}(0) = \alpha,\ \bar{u}'(0) = \beta\,. \tag{73}$$

Here we have four dimensionless variables

$$\alpha = \frac{I}{u_c}, \tag{74}$$

$$\beta = \frac{Vt_c}{u_c} = \frac{V}{\omega u_c}, \tag{75}$$

$$\gamma = \frac{t_c^2 A}{mu_c} = fracAku_c, \tag{76}$$

$$\delta = \frac{t_c}{\psi^{-1}} = \frac{\psi}{\omega}\,. \tag{77}$$

These dimensionless variables have interpretations as ratios of physical effects:

- α: ratio of the initial displacement and the typical displacement u_c,
- β: ratio of the initial velocity and the typical velocity measure u_c/t_c,
- γ: ratio of the forcing A and the mass times acceleration mu_c/t_c^2 *or* the ratio of the forcing and the spring force ku_c
- δ: ratio of the frequencies or the time scales of the forcing and the free vibrations.

Software. Any solver for (71) can be used for (73). In particular, we may use the `solver` function in the file `vib.py`[13]:

```
def solver(I, V, m, b, s, F, dt, T, damping='linear'):
```

for solving

$$mu'' + f(u') + s(u) = F(t), \quad u(0) = I,\ u'(0) = V,\ t \in (0, T],$$

with time steps `dt`. For `damping='linear'`, we have $f(u') = bu'$. The other value is `'quadratic'`, meaning $f(u') = b|u'|u'$. Given α, β, γ, and δ, and appropriate call is

```
u, t = solver(I=alpha, V=beta, m=1, b=0,
              s=lambda u: u, F=lambda t: gamma*cos(delta*t),
              dt=2*pi/n, T=2*pi*P)
```

where `n` is the number intervals per period and `P` is the number of periods to be simulated.

We way wrap this call in a `solver_scaled` function as we did in Section 2.2, but this is done in Section 3.4.

[13] `http://tinyurl.com/nm5587k/vib/vib.py`

Choice of u_c close to resonance. We have chosen not to specify u_c in the formulas above. Now we shall discuss various choices of u_c. Close to resonance, when $\psi \sim \omega$, we may set $u_c = A/(m(\omega^2 - \psi^2))$. The dimensionless numbers become in this case

$$\begin{aligned}
\alpha &= \frac{I}{u_c} = \frac{I}{A/k}(1-\delta^2), \\
\beta &= \frac{V}{\omega u_c} = \frac{V\sqrt{km}}{A}(1-\delta^2), \\
\gamma &= \frac{A}{ku_c} = 1-\delta^2, \\
\delta &= \frac{\psi}{\omega}\,.
\end{aligned}$$

With $\psi = 0.99\omega$, $\delta = 0.99$, $V = 0$, $\alpha = \gamma = 1 - \delta^2 = 0.02$, we have the problem

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \bar{u} = 0.02\cos(0.99\bar{t}), \quad \bar{u}(0) = 0.02,\ \bar{u}'(0) = 0\,.$$

This is a problem with a very small initial condition and a very small forcing, but the state close to resonance brings the amplitude up to about unity, see the result of numerical simulations with $\delta = 0.99$ in Figure 7. Neglecting α, the solution is given by (72), which here means $A = 1 - \delta^2$, $m = 1$, $\omega = 1$, $\psi = \delta$:

$$\bar{u}(\bar{t}) = 2\sin(-0.005\bar{t})\sin(0.995\bar{t})\,.$$

Note that this is a problem which demands very high accuracy in the numerical calculations. Using 20 time steps per period gives a significant phase error and an amplitude of about 1.4. We used 160 steps per period for the results in Figure 7.

Unit size of all terms in the ODE. Using the displacement scale $u_c = A/k$ leads to (73) with

$$\begin{aligned}
\alpha &= \frac{I}{u_c} = \frac{I}{A/k}, \\
\beta &= \frac{V}{\omega u_c} = \frac{Vk}{A\omega}, \\
\gamma &= \frac{A}{ku_c} = 1, \\
\delta &= \frac{\psi}{\omega}\,.
\end{aligned}$$

Simulating a case with $\delta = 0.5$, $\alpha = 1$, and $\beta = 0$ gives the oscillations in Figure 8, which is a case away from resonance, and the amplitude is about unity. However, choosing $\delta = 0.99$ (close to resonance) results in a figure similar to Figure 7, except that the amplitude is about 10^2 because of the moderate size of u_c. The present scaling is therefore most suitable away from resonance, and when the terms containing $\cos\omega t$ and $\sin\omega t$ are important (e.g., $\omega \gg \psi$).

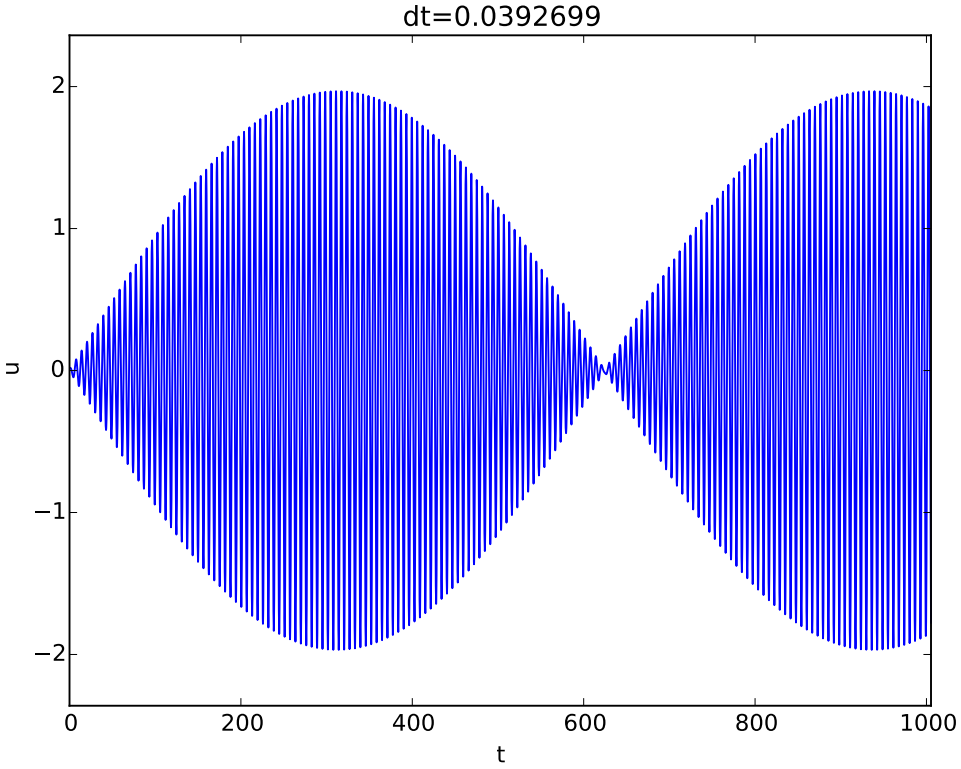

Figure 7: Forced undamped vibrations close to resonance.

Choice of u_c when $\psi \gg \omega$. Finally, we may look at the case where $\psi \gg \omega$ such that $u_c = A/(m\psi^2)$ is a relevant scale (i.e., omitting ω^2 compared to ψ^2 in the denominator), but in this case we should use $t_c = 1/\psi$ since the variations in the force are much greater than in the free vibrations of the system. This choice of t_c changes the scaled ODE to

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \delta^{-2}\bar{u} = \gamma\cos(\bar{t}), \quad \bar{u}(0) = \alpha,\ \bar{u}'(0) = \beta, \tag{78}$$

where

$$\begin{aligned}
\alpha &= \frac{I}{u_c} = \frac{I}{A/k}\delta^2, \\
\beta &= \frac{Vt_c}{u_c} = \frac{V\sqrt{km}}{A}\delta, \\
\gamma &= \frac{t_c^2 A}{mu_c} = 1, \\
\delta &= \frac{t_c}{\psi^{-1}} = \frac{\psi}{\omega}.
\end{aligned}$$

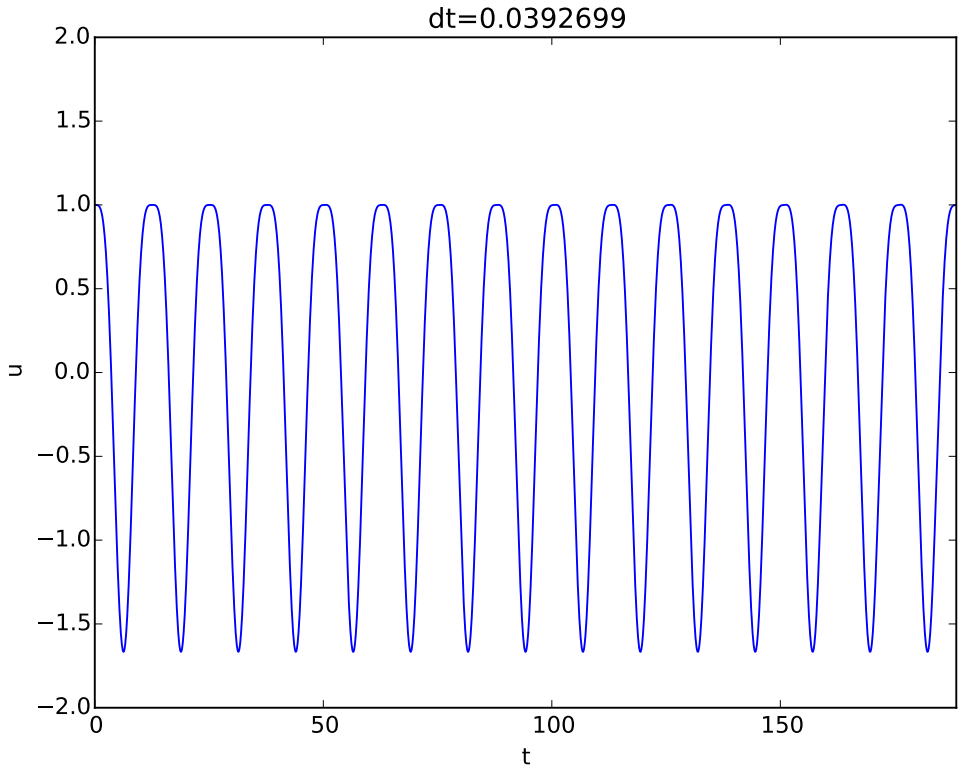

Figure 8: Forced undamped vibrations away from resonance.

In the regime $\psi \gg \omega$, $\delta \gg 1$, thus making α and β large. However, if α and/or β is large, the initial condition dominates over the forcing, and will also dominate the amplitude of u, thereby making the scaling of u inappropriate. In case $I = V = 0$ so that $\alpha = \beta =$, (72) predicts ($A = m = 1$, $\omega = \delta^{-1}$, $\psi = 1$)

$$\bar{u}(\bar{t}) = (\delta^{-2} - 1)^{-1} 2 \sin\left(\frac{1}{2}(1 - \delta^{-1})\bar{t}\right) \sin\left(\frac{1}{2}(1 + \delta^{-1})\bar{t}\right),$$

which has an amplitude about 2 for $\delta \gg 1$, i.e., very small amplitude. Figure 9 shows numerical simulations with 160 time steps per period of $F(t)$.

With $\alpha = 0.05\delta^2 = 5$, we get a significant contribution from the free vibrations (the homogeneous solution of the ODE) as shown in Figure 10. For larger α values, one must base

Displacement scale based on I. Choosing $u_c = I$ gives

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \bar{u} = \gamma \cos(\delta\bar{t}), \quad \bar{u}(0) = 1, \ \bar{u}'(0) = \beta, \tag{79}$$

with

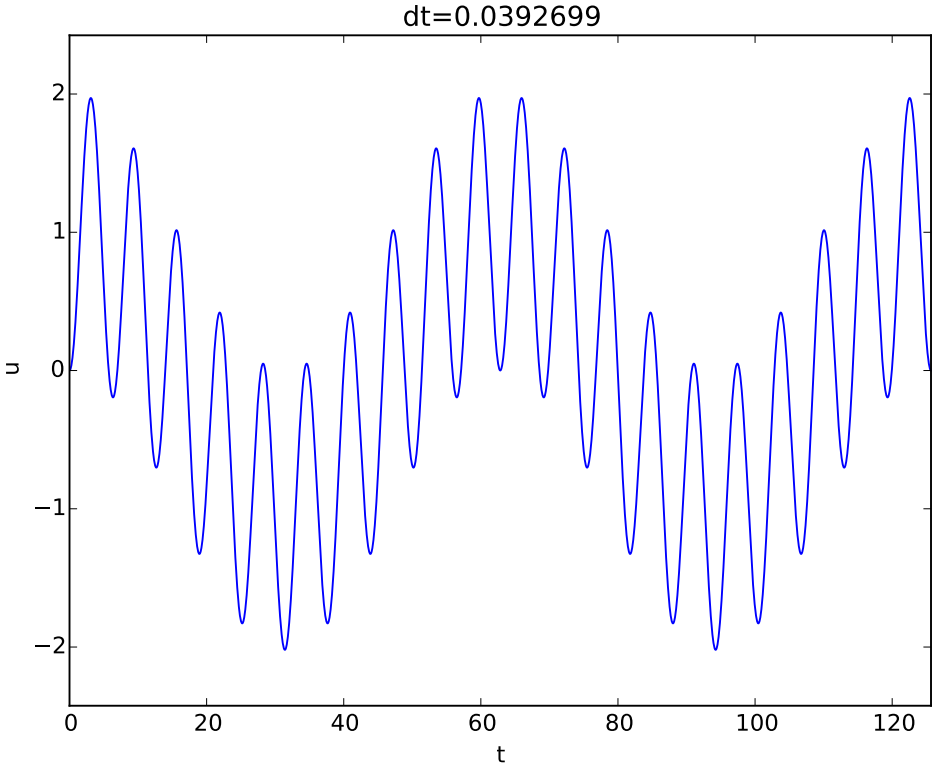

Figure 9: Forced undamped vibrations with rapid forcing.

$$\beta = \frac{V}{I}\sqrt{\frac{mk}{=}}\frac{Vt_c}{u_c}, \gamma \qquad = \frac{A}{kI} = \frac{tc^2 A}{mu_c} = \frac{A}{ku_c}\,. \tag{80}$$

This scaling is not relevant close to resonance since then $u_c \gg I$.

3.4 Damped vibrations with forcing

We now introduce a linear damping force $bu'(t)$ in the equation of motion:

$$mu'' + bu' + ku = A\cos(\psi t), \quad u(0) = I,\ u'(0) = V\,. \tag{81}$$

Scaling results in

$$\frac{d^2\bar{u}}{d\bar{t}^2} + \frac{t_c b}{m}\frac{d\bar{u}}{d\bar{t}} + \frac{t_c^2 k}{m}\bar{u} = \frac{t_c^2}{mu_c}A\cos(\psi t_c\bar{t}), \quad \bar{u}(0) = \frac{I}{u_c},\ \bar{u}'(0) = \frac{Vt_c}{u_c}\,. \tag{82}$$

The exact solution. To choose scales, it is a great advantage to look into exact solutions. Using SymPy to solve (81) turns out to be difficult, because with damping, we have three solutions of (81), and for vibrations we are only interested in one of them, the one with damped oscillations.

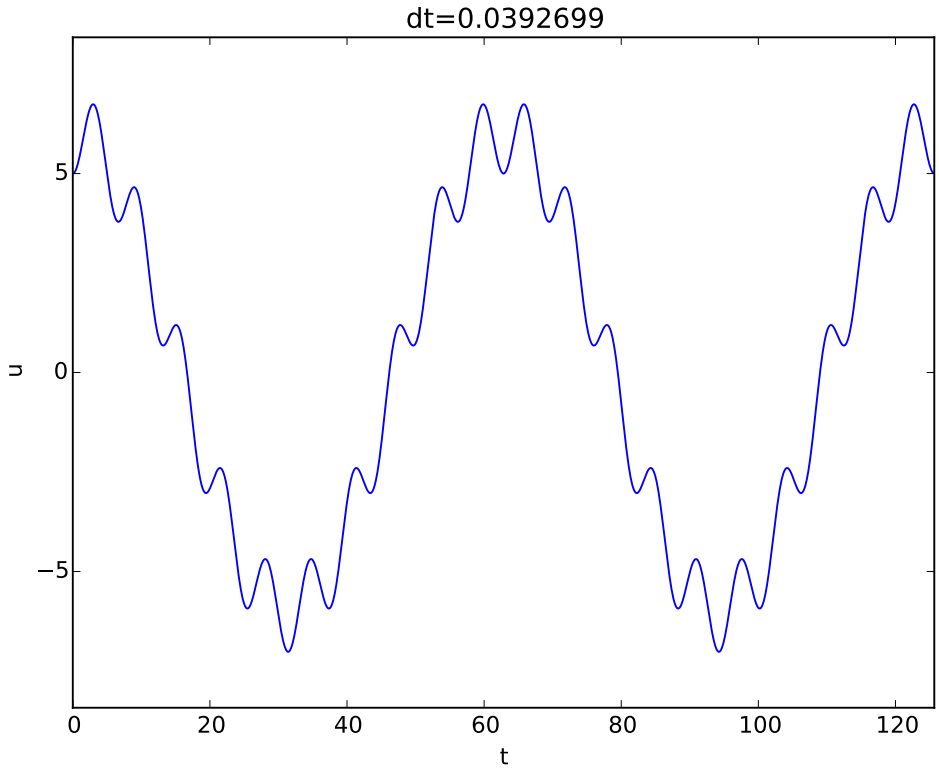

Figure 10: Forced undamped vibrations with rapid forcing and initial displacement of 5.

So we have to drop `dsolve` and perform manual steps, either by hand or with the aid of SymPy. The solution is composed of a homogeneous solution u_h of $mu'' + bu' + ku = 0$ and one particular solution u_p of the nonhomogeneous equation $mu'' + bu' + ku = A\cos(\psi t)$. The homogeneous solution with damped oscillations (requiring $b < 2\sqrt{mk}$) can be found by the following code. We have divided the differential equation by m and introduced $B = \frac{1}{2}b/m$ and let `A1` represent A/m to simplify expressions and help SymPy with less symbols in the equation. Without these simplifications, SymPy stalls in the computations due to too many symbols in the equation.

```
u = symbols('u', cls=Function)
t, w, B, A, A1, m, psi = symbols('t w B A A1 m psi',
                                 positive=True, real=True)

def ode(u, homogeneous=True):
    h = diff(u, t, t) + 2*B*diff(u, t) + w**2*u
    f = A1*cos(psi*t)
    return h if homogeneous else h - f

# Find coefficients in polynomial (in r) for exp(r*t) ansatz
r = symbols('r')
ansatz = exp(r*t)
poly = simplify(ode(ansatz)/ansatz)

# Convert to polynomial to extract coefficients
```

```
poly = Poly(poly, r)
# Extract coefficients in poly: a_*t**2 + b_*t + c_
a_, b_, c_ = poly.coeffs()
# Assume b_**2 - 4*a_*c_ < 0
d = -b_/(2*a_)
if a_ == 1:
    omega = sqrt(c_ - (b_/2)**2)  # nicer formula
else:
    omega = sqrt(4*a_*c_ - b_**2)/(2*a_)

# The homogeneous solution is a linear combination of a
# cos term (u1) and a sin term (u2)
u1 = exp(d*t)*cos(omega*t)
u2 = exp(d*t)*sin(omega*t)
C1, C2, V, I = symbols('C1 C2 V I', real=True)
u_h = simplify(C1*u1 + C2*u2)
print 'u_h:', u_h
```

The print out shows

$$u_h = e^{-Bt}\left(C_1\cos(\sqrt{\omega^2 - B^2}t) + C_2\sin(\sqrt{\omega^2 - B^2}t)\right),$$

where C_1 and C_2 must be determined by the initial conditions later. It is wise to check that u_h is indeed a solution of the homogeneous differential equation:

```
assert simplify(ode(u_h)) == 0
```

The ansatz for the particular solution u_p is

$$u_p = C_3\cos(\psi t) + C_4\sin(\psi t),$$

which inserted in the ODE gives two equations for C_3 and C_4. The relevant SymPy statements are

```
# Particular solution
C3, C4 = symbols('C3 C4')
u_p = C3*cos(psi*t) + C4*sin(psi*t)
eqs = simplify(ode(u_p, homogeneous=False))

# Collect cos(omega*t) terms
print 'eqs:', eqs
eq_cos = simplify(eqs.subs(sin(psi*t), 0).subs(cos(psi*t), 1))
eq_sin = simplify(eqs.subs(cos(psi*t), 0).subs(sin(psi*t), 1))
s = solve([eq_cos, eq_sin], [C3, C4])
u_p = simplify(u_p.subs(C3, s[C3]).subs(C4, s[C4]))

# Check that the solution is correct
assert simplify(ode(u_p, homogeneous=False)) == 0
```

Using the initial conditions for the complete solution $u = u_h + u_p$ determines C_1 and C_2:

```
u_sol = u_h + u_p  # total solution
# Initial conditions
eqs = [u_sol.subs(t, 0) - I, u_sol.diff(t).subs(t, 0) - V]
# Determine C1 and C2 from the initial conditions
s = solve(eqs, [C1, C2])
u_sol = u_sol.subs(C1, s[C1]).subs(C2, s[C2])
```

Finally, we should check that `u_sol` is indeed the correct solution:

```
checks = dict(
    ODE=simplify(expand(ode(u_sol, homogeneous=False))),
    IC1=simplify(u_sol.subs(t, 0) - I),
    IC2=simplify(diff(u_sol, t).subs(t, 0) - V))
for check in checks:
    msg = '%s residual: %s' % (check, checks[check])
    assert checks[check] == sympify(0), msg
```

Finally, we may take `u_sol = u_sol.subs(A, A/m)` to get the right expression for the solution. Using `latex(u_sol)` results in a huge expression, which should be manually ordered to something like the following:

$$
\begin{aligned}
u =& \frac{Am^{-1}}{4B^2\psi^2 + \Omega^2}\left(2B\psi\sin\left(\psi t\right) - \Omega\cos\left(\psi t\right)\right) + \\
& e^{-Bt}\left(C_1\cos\left(t\sqrt{\omega^2 - B^2}\right) + C_2\sin\left(t\sqrt{\omega^2 - B^2}\right)\right) \\
C_1 =& \frac{Am^{-1}\Omega + 4IB^2\psi^2 + I\Omega^2}{4B^2\psi^2 + \Omega^2} \\
C_2 =& \frac{-Am^{-1}B\Omega + 4IB^3\psi^2 + IB\Omega^2 + 4VB^2\psi^2 + V\Omega^2}{\sqrt{\omega^2 - B^2}\left(4B^2\psi^2 + \Omega^2\right)}, \\
\Omega =& \psi^2 - \omega^2\,.
\end{aligned}
$$

The most important feature of this solution is that there are two time scales with frequencies ψ and $\sqrt{\omega^2 - B^2}$, but the latter appears in terms that decay as e^{Bt} in time. The attention is usually on longer periods of time, so in that case the solution simplifies to

$$
\begin{aligned}
u &= \frac{Am^{-1}}{4B^2\psi^2 + \Omega^2}\left(2B\psi\sin\left(\psi t\right) - \Omega\cos\left(\psi t\right)\right) \\
&= \frac{A}{m}\frac{1}{\sqrt{4B^2\psi^2 + \Omega^2}}\cos(\psi t + \phi)\frac{(\psi\omega)^{-1}}{(\psi\omega)^{-1}} \\
&= \frac{A}{k}Q\delta^{-1}\left(1 + Q^2(\delta - \delta^{-1})\right)^{-\frac{1}{2}}\cos(\psi t + \phi),
\end{aligned} \tag{83}
$$

where we have introduced the dimensionless numbers

$$
Q = \frac{\omega}{2B}, \quad \delta = \frac{\psi}{\omega},
$$

and

$$
\phi = \tan^{-1}\left(-\frac{2B}{\omega^2 - \psi^2}\right) = \tan^{-1}\left(\frac{Q^{-1}}{\delta^2 - 1}\right)\,.
$$

Q is commonly called *quality factor* and ϕ is the *phase shift*. Dividing (83) by A/k, which is a common scale for u, gives the dimensionless relation

$$
\frac{u}{A/k} = \frac{Q}{\delta}R(Q,\delta)^{\frac{1}{2}}\cos(\psi t + \phi), \quad R(Q,\delta) = \left(1 + Q^2(\delta - \delta^{-1})\right)^{-1}\,. \tag{84}
$$

Choosing scales. Much of the discussion about scales in the previous sections are relevant also with damping is included. Although the oscillations with frequency $\sqrt{\omega^2 - B^2}$ die out for $t \gg B^{-1}$, we start with using this frequency for the time scale. A highly relevant assumption for engineering applications of (81) is that the damping is small. Therefore, $\sqrt{\omega^2 - B^2}$ is close to ω and we simply apply $t_c = 1/\omega$ as before.

The coefficient in front of the $\bar{u}'$ term then becomes

$$\frac{b}{m\omega} = \frac{2B}{\omega} = Q^{-1}\,.$$

Choice of u_c at resonance. The relevant scale for u_c at or nearby resonance ($\psi = \omega$) becomes different from the previous section, since with damping, the maximum amplitude a finite value. For $t \gg 1/B$ when the $\sin\psi t$ term is dominating, we have for $\psi = \omega$:

$$u = \frac{Am^{-1}2B\psi}{4B^2\psi^2}\sin(\psi t) = \frac{A}{2Bm\psi}\sin(\psi t) = \frac{A}{b\psi}\sin(\psi t)\,.$$

This motivates the choice

$$u_c = \frac{A}{b\psi} = \frac{A}{b\omega}\,.$$

(It is wise during computations like this to stop and check the dimensions: A must be $[\mathrm{MLT}^{-2}]$ from the original equation ($F(t)$ must have same dimension as mu''), bu' also has dimension $[\mathrm{MLT}^{-2}]$, implying that b has dimension $[\mathrm{MT}^{-1}]$. A/b then has dimension LT^{-1}, and $A/(b\psi)$ gets dimension $[L]$, which matches what we want for u_c.)

The differential equation on dimensionless form becomes

$$\frac{d^2\bar{u}}{d\bar{t}^2} + Q^{-1}\frac{d\bar{u}}{d\bar{t}} + \bar{u} = \gamma\cos(\delta\bar{t}), \quad \bar{u}(0) = \alpha,\ \bar{u}'(0) = \beta, \tag{85}$$

with

$$\alpha = \frac{I}{u_c} = \frac{Ib}{A}\sqrt{\frac{k}{m}}, \tag{86}$$

$$\beta = \frac{Vt_c}{u_c} = \frac{Vb}{A}, \tag{87}$$

$$\gamma = \frac{t_c^2 A}{mu_c} = \frac{b\omega}{k}, \tag{88}$$

$$\delta = \frac{t_c}{\psi^{-1}} = \frac{\psi}{\omega} = 1\,. \tag{89}$$

Choice of u_c when $\omega \gg \psi$. In the limit $\omega \gg \psi$ and $t \gg 1/B$,

$$u \approx \frac{A}{m\omega^2}\cos\psi t = \frac{A}{k}\cos\psi t,$$

showing that $u_c = A/k$ is an appropriate displacement scale. (Alternatively, we get this scale also from $\gamma = 1$ in the ODE.) The dimensionless numbers α, β, and δ are as for the forced vibrations without damping.

Choice of u_c when $\omega \ll \psi$. In the limit $\omega \ll \psi$, we should base t_c on the rapid variations in the excitation, $t_c = 1/\psi$.

3.5 Oscillating electric circuits

The differential equation for an oscillating electric circuit is very similar to the equation for forced, damped, mechanical vibrations, and their dimensionless form is identical. The current $I(t)$ in a circuit with an inductor with inductance L, a capacitor with capacitance C, and overall resistance R, obeys the equation

$$\ddot{I} + \frac{R}{L}\dot{I} + \frac{1}{LC}I = \dot{V}(t), \tag{90}$$

where $V(t)$ is the voltage source powering the circuit. We introduce

$$\bar{I} = \frac{I}{I_c}, \quad \bar{t} = \frac{t}{t_c},$$

and get

$$\frac{d^2\bar{I}}{d\bar{t}^2} + \frac{t_c R}{L}\frac{d\bar{I}}{d\bar{t}} + \frac{t_c^2}{LC}\bar{I} = \frac{t_c^2 V_c}{I_c}\bar{V}(t)\,.$$

Here, we have scaled $V(t)$ according to

$$\bar{V}(bart) = \frac{V(t_c\bar{t})}{\max_t V(t)}\,.$$

The time scale t_c is chosen to make $\ddot{I}$ and $I/(LC)$ balance, $t_c = \sqrt{LC}$. Choosing I_c to make the coefficient in the source term of unit size, means $I_c = LCV_c$. With

$$\alpha = \frac{R}{2}\sqrt{\frac{C}{L}},$$

we get the scaled equation

$$\frac{d^2\bar{I}}{d\bar{t}^2} + 2\alpha\frac{d\bar{I}}{d\bar{t}} + \bar{I} = \bar{V}(t)\,. \tag{91}$$

Two additional dimensionless variables will arise from the initial conditions for I.

4 The wave equation

A standard, linear, one-dimensional wave equation problem in a homogeneous medium may be written as

$$\frac{\partial^2 u}{\partial t^2} = c^2\frac{\partial^2 u}{\partial x^2}, \quad x \in (0, L),\ t \in (0, T], \tag{92}$$

where c is the constant wave velocity of the medium. With a briefer notation, where subscripts indicate derivatives, the PDE (92) can be written $u_{tt} = c^2 u_{xx}$. This subscript notation will occasionally be used later.

In multi dimensions in heterogeneous media we have the generalization

$$\frac{\partial^2 u}{\partial t^2} = \nabla\cdot\left(c^2\nabla u\right) + f, \quad x,y,z\in\Omega,\ t\in(0,T]\,. \tag{93}$$

How to scale time depends on the PDE, the spatial scale depends on the domain, and the scale of u usually depends on the type of boundary and initial condition.

4.1 Simple homogeneous Dirichlet conditions

Let us first start with homogeneous Dirichlet conditions in space and no initial velocity u_t:

$$u(x,0) = I(x), \qquad x\in[0,L], \tag{94}$$
$$\frac{\partial}{\partial t}u(x,0) = 0, \qquad x\in[0,L], \tag{95}$$
$$u(0,t) = 0, \qquad t\in(0,T], \tag{96}$$
$$u(L,t) = 0, \qquad t\in(0,T]. \tag{97}$$

The independent variables are x and t, while u is the dependent variable. The rest of the parameters, c, L, T, and $I(x)$, are given data.

We start with introducing dimensionless versions of the independent and dependent variables:

$$\bar{x} = \frac{x}{x_c}, \quad \bar{t} = \frac{t}{t_c}, \quad \bar{u} = \frac{u}{u_c}\,.$$

Inserting the $x = x_c\bar{x}$, etc., in (92) and (94)-(97) gives

$$\frac{\partial^2\bar{u}}{\partial\bar{t}^2} = \frac{t_c^2c^2}{x_c^2}\frac{\partial^2\bar{u}}{\partial x^2}, \qquad \bar{x}\in(0,L/x_c),\ \bar{t}\in(0,T/t_c],$$
$$\bar{u}(\bar{x},0) = \frac{I(x_c\bar{x})}{u_c}, \qquad \bar{x}\in[0,L/x_c],$$
$$\frac{\partial}{\partial\bar{t}}\bar{u}(\bar{x},0) = 0, \qquad \bar{x}\in[0,L/x_c],$$
$$\bar{u}(0,\bar{t}) = 0, \qquad \bar{t}\in(0,T/t_c],$$
$$\bar{u}(L/x_c,\bar{t}) = 0, \qquad \bar{t}\in(0,T/t_c].$$

The key question is how to define the scales. A natural choice is $x_c = L$ since this makes $\bar{x}\in[0,1]$. For the problem governed by (92) we have some analytical insight, namely that the solution behaves like

$$u(x,t) = f_R(x-ct) + f_R(x+ct), \tag{98}$$

i.e., a right- and left-going wave with velocity c. The initial conditions constrain the choices of f_R and f_L to $f_L + f_R = I$ and $-cf_L' + cf_R' = 0$. The solution is $f_R = f_L = \frac{1}{2}$, and consequently

$$u(x,t) = \frac{1}{2}I(x-ct) + \frac{1}{2}I(x+ct),$$

which tells that the initial condition splits in two, half of it moves to the left and half to the right. This means in particular that we can choose $u_c = \max_x|I(x)|$ and get $|\bar{u}|\leq 1$, which is a goal.

Regarding the time scale, we may look at the two terms in the scaled PDE and argue that if $|u|$ and its derivatives are to be of order unity, then the size of the second-order derivatives

should be the same, and t_c can be chosen to make the coefficient $t_c^2c^2/x_c^2$ unity, i.e., $t_c = L/c$. Another reasoning may set t_c as the time it takes the wave to travel through the domain $[0, L]$. Since the wave has constant speed c, $t_c = L/c$.

With the described choices of scales, we end up with the dimensionless initial-boundary value problem

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{\partial^2 \bar{u}}{\partial x^2}, \qquad \bar{x} \in (0,1),\ \bar{t} \in (0,\bar{T}], \tag{99}$$

$$\bar{u}(\bar{x},0) = \frac{I(\bar{x}L)}{\max_{x\in(0,L)} |I(x)|}, \qquad \bar{x} \in [0,1], \tag{100}$$

$$\frac{\partial}{\partial \bar{t}}\bar{u}(\bar{x},0) = 0, \qquad \bar{x} \in [0,1], \tag{101}$$

$$\bar{u}(0,\bar{t}) = 0, \qquad \bar{t} \in (0,\bar{T}], \tag{102}$$

$$\bar{u}(1,\bar{t}) = 0, \qquad \bar{t} \in (0,\bar{T}]. \tag{103}$$

Here, $\bar{T} = Tc/L$.

The striking feature of (99)-(103) is that there are *no physical parameters* involved! Everything we need to specify is the shape of the initial condition and then scale it such that it is less than or equal to 1.

The physical solution with dimension is recovered from $\bar{u}(\bar{x},\bar{t})$ through

$$u(x,t) = \max_{x\in(0,L)} I(x)\, \bar{u}(\bar{x}L, \bar{t}L/c) \tag{104}$$

4.2 Implementation of the scaled wave equation

How do we implement (99)-(103)? As for the simpler mathematical models, I suggest to implement the model with dimensions and observe how to set parameters to obtain the scaled model. In the present case, one must choose $L = 1$, $c = 1$, and scale I by its maximum value. That's all!

Several implementations of 1D wave equation models with different degree of mathematical and software complexity appear in the directory `wave/wave1D`[14]. The simplest version is `wave1D_u0.py`[15] that implements (92) and (94-(97). This is the code to be used in the following. It is described in Sections **??** in [2].

Waves on a string. As example, we may let the original initial-boundary value problem (92)-(97) model vibrations of a string on a string instrument. With u as the displacement of the string, the boundary conditions $u = 0$ at the ends are relevant, as well as the zero velocity condition $\partial u/\partial t = 0$ at $t = 0$. The initial condition $I(x)$ has typically a triangular shape for a picked guitar string. The physical problem needs parameters for the amplitude of $I(x)$, the length L of the string, and the value of c for the string. Only the latter is challenging as it involves relating c to the pitch (i.e., time frequency) of the string. In the scaled problem, we can forget about all this. We simply set $L = 1$, $c = 1$, and let $I(x)$ have a peak of unity at $x = x_0 \in (0,1)$:

$$\frac{I(x)}{\max_x I(x)} = \begin{cases} x/x_0, & x < x_0, \\ (1-x)/(1-x_0), & \text{otherwise} \end{cases}$$

The dimensionless coordinate of the peak, x_0, is the only dimensionless parameter in the problem. For fixed x_0, one single simulation will capture all possible solutions with such a triangular shape.

[14] `http://tinyurl.com/nm5587k/wave/wave1D`

[15] `http://tinyurl.com/nm5587k/wave/wave1D/wave1D_u0.py`

Detecting an already computed case. In Section 2.2 we demonstrated the use of `joblib` for making a function that detects if a case has already been run and in that case the previous solution can be returned from a database. It turns out that `joblib` cannot handle functions with function arguments, which we have a lot of in the `solver` functions for 1D wave equations.

A manual strategy taken from `wave1D_dn_vc.py`[16] and explained in Section **??** in [4] is to convert all input data to the `solver` function to a string, which is thereafter converted to an SHA1 hash string (via `hashlib.sha1`) and used to recognize the input. A SHA1 string is also suitable as part of a file or directory name where computed solutions can be stored.

We can, in the wave equation solver retrieve the solution, rather than computing it, if the hash string is the same (because then the computations have already been done). This can save a lot of computations if a scaled solution can be reused in a number of cases with dimensions. We will sketch the code that implements the idea.

A solver for the scaled problem is first developed. We limit the focus to the simple constant-coefficient wave equation with $u_t(x,0) = 0$. The solver for the unscaled problem is taken from the previously mentioned `wave1D_u0.py` file.

```
# Enable loading modules in the wave eq solver and softeng2 dirs
sys.path.insert(0, os.path.join(
    os.pardir, os.pardir, 'wave', 'src-wave', 'wave1D'))
sys.path.insert(0, os.path.join(
    os.pardir, os.pardir, 'softeng2', 'src-softeng2'))
from wave1D_u0 import solver as solver_unscaled
from Storage import Storage

def solver_scaled(I, dt, C, T):
    """
    Solve 1D wave equation in dimensionless form.
    """
    # Make a hash of the arguments
    import inspect, hashlib
    data = inspect.getsource(I) + '_' + str(dt) + '_' + \
           str(C) + '_' + str(T)
    # Not fool proof: if x0 changes value, I source is the same...
    hashed_input = hashlib.sha1(data).hexdigest()

    cachedir = 'tmp_%s' % hashed_input
    is_computed = os.path.isdir(cachedir)
    print 'cachedir:', cachedir, is_computed
    storage = Storage(cachedir, verbose=0)

    def action(u, x, t, n):
        if n == 0:
            storage.save('x', x)
            storage.save('t', t)
        storage.save('u%d' % n, u)

    if is_computed:
        print 'No need to compute the numerical solution'
        return storage
    else:
        print 'Computing the numerical solution'
        solver_unscaled(
            I=I, V=0, f=0, c=1, L=1, dt=dt, C=C, T=T,
            user_action=action)
        return storage
```

This function employs ideas described in Sections **??** and **??** in [4] for storing arrays on disk with use of `joblib` (class `Storage`) and recognizing previous input through a hash string. If the

[16] `http://tinyurl.com/nm5587k/wave/wave1D/wave1D_dn_vc.py`

input is the same, the hash is the same and we can test on the existence of a directory whose name contains the hash. If that directory exists, the solution for this set of input data is already computed, and we can just return the `storage` object from which one can retrieve the space and time mesh as well as all the solutions `u0`, `u1`, and so on.

A specific application of this simple solver is the vibrations of a guitar string. The scaled version depends only on C (if we say T is fixed and N_x is fixed through Δt). The string vibrations can be simulated by the following function:

```
def guitar_scaled(C, animate=True):
    """Triangular wave (pulled guitar string)."""
    L = 1.0
    x0 = 0.8*L
    T = 2
    Nx = 50; dx = L/float(Nx)
    dt = dx/1  # Choose dt at the stability limit

    def I(x):
        return x/x0 if x < x0 else (L-x)/(L-x0)

    storage = solver_scaled(I, dt, C, T)
    if not animate:
        return storage

    from scitools.std import plot
    x = storage.retrieve('x')
    t = storage.retrieve('t')
    for n in range(len(t)):
        u = storage.retrieve('u%d' %n)
        plot(x, u, 'r-', label='t=%.2f' % t[n],
             axis=[x[0], x[-1], -1.2, 1.2])
    return storage
```

Although the partial differential equation model has no physical parameters (assuming x_0 fixed), the corresponding numerical model depends on the Courant number $C = c\Delta t/\Delta x$ and the length T of the simulations.

To solve an unscaled problem, we need some unscale functions:

```
def unscale_u(u, I_max, c, L):
    return I_max*u

def unscale_x(x, I_max, c, L):
    return x*L

def unscale_t(t, I_max, c, L):
    return t*L**2/float(c)
```

We can now easily solve a range of unscaled cases by

```
def guitar(C, I_max, c, L):
    """Triangular wave (pulled guitar string). Unscaled version."""
    storage = guitar_scaled(C, animate=False)
    x = storage.retrieve('x')
    t = storage.retrieve('t')
    x = unscale_x(x, I_max, c, L)
    t = unscale_t(t, I_max, c, L)

    from scitools.std import plot
    for n in range(len(t)):
        u = storage.retrieve('u%d' %n)
        u = unscale_u(u, I_max, c, L)
```

```
        plot(x, u, 'r-', label='t=%.2f' % t[n],
             axis=[x[0], x[-1], -I_max*1.2, 1.2*I_max])
```

If `guitar_scaled` figures out that the scaled problem is already solved, it just returns the `storage` object, otherwise it performs calculations. Anyway, we retrieve the space and time mesh as well as all the solutions. The `plot` function from SciTools[17] is used for compact code for animation, but Matplotlib can equally well be used (with a bit more coding).

Suppose we run three calls to `guitar` with three different values of `I_max`. The output will be

```
Computing the numerical solution
No need to compute the numerical solution
No need to compute the numerical solution
```

This indicates that we rely on the scaled solution for the two other cases with different `I_max` parameter. Running such a program again will avoid all computations and show movies solely based on precomputed file data.

hpl 3: I make a big issue of retrieving data from file, but in more complicated cases, this has little practical value as the number of dimensionless numbers gets large (approaching the number of parameters anyway). In the present simple cases, the computations are so fast that there is little to gain by avoiding them. The text is therefore biased with respect to the practical benefit of reusing old solutions. Nevertheless, it probably does not hurt to document the idea? Need views on this.

4.3 Time-dependent Dirichlet condition

A generalization of (92)-(97) is to allow for a time-dependent Dirichlet condition at one end, say $u(0,t) = U_L(t)$. At the other end we may still have $u = 0$. This new condition at $x = 0$ may model a specified wave that enters the domain. For example, if we feed in a monochromatic wave $A\sin(k(x - ct))$ from the left end, $U_L(t) = A\sin(kct)$. This forcing of the wave motion has its own amplitude and time scale that could affect the choice of u_c and t_c.

The main difference from the previous initial-boundary value problem is the condition at $x = 0$, which now reads

$$\bar{u}(0,\bar{t}) = \frac{U_L(\bar{t}t_c)}{u_c}$$

in scaled form.

Regarding the characteristic time scale, it is natural to base this scale on the wave propagation velocity and not on the time scale of $U_L(t)$, because the time scale of U_L basically determines whether short or long waves are fed in at the boundary. All waves, long or short, propagate with the same velocity c. We therefore continue to use $t_c = L/c$.

The solution u will have one wave contribution from the initial condition I and one from the feeding of waves at $x = 0$. This gives us three choices of u_c: $\max_x |I| + \max_t |U_L|$, $\max_x |I|$, or $\max_t |U_L|$. The first seems relevant if the size of I and U_L are about the same, but then we can choose either $\max_x |I|$ or $\max_t |U_L|$ as characteristic size of u since a factor of 2 is not important. If I is much less than U_L, $u_c = \max_t |u_L|$ is relevant, while $u_c = \max_x I$ is the choice when I has much bigger impact than U_L on u.

With $u_c = \max_t |U_L(t)|$, we get the scaled problem

[17] `https://github.com/hplgit/scitools`

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{\partial^2 \bar{u}}{\partial \bar{x}^2}, \qquad \bar{x} \in (0,1),\ \bar{t} \in (0,\bar{T}], \tag{105}$$

$$\bar{u}(\bar{x},0) = \frac{I(x_c\bar{x})}{\max_t |U_L(t)|}, \qquad \bar{x} \in [0,1], \tag{106}$$

$$\frac{\partial}{\partial \bar{t}}\bar{u}(\bar{x},0) = 0, \qquad \bar{x} \in [0,1], \tag{107}$$

$$\bar{u}(0,\bar{t}) = \frac{U_L(\bar{t}t_c)}{\max_t |U_L(t)|}, \qquad \bar{t} \in (0,\bar{T}], \tag{108}$$

$$\bar{u}(1,\bar{t}) = 0, \qquad \bar{t} \in (0,\bar{T}]. \tag{109}$$

Also this problem is free of physical parameters like c and L. The input is completely specified by the shape of $I(x)$ and $U_L(t)$.

Software for the original problem with dimensions can be reused for (105)-(109) by setting $L = 1$, $c = 1$, and scaling $U_L(t)$ and $I(x)$ by $\max_t |U_L(t)|$.

As an example, consider

$$U_L(t) = a\sin(\omega t) \text{ for } 0 \leq t \leq 2\frac{\omega}{2\pi}, \text{ else } 0,$$
$$I(x) = Ae^{-(x-L/2)^2/\sigma^2}.$$

That is, we start with a Gaussian peak-shaped wave in the center of the domain and feed in a sinusoidal wave at the left end for two periods. The solution will be the sum of three waves: two parts from the initial condition, plus the wave fed in from the left.

Since $\max_t |U_L| = a$ we get

$$\bar{u}(\bar{x},0) = \frac{A}{a}e^{-(L/\sigma)^2(\bar{x}-\frac{1}{2})^2},$$
$$\bar{u}(0,\bar{t}) = \sin(\bar{t}\omega L/c).$$

Here, U_L models an incoming wave $a\sin(k(x-ct)$, with k specified (makes waves of length $\lambda = 2\pi/k$), we have $\omega = kc$, and $\bar{u}(0,\bar{t}) = \sin(kL\bar{t}) = \sin(2\pi\bar{t}L/\lambda)$. (This formula demonstrates the previous assertion that the time scale of U_L, i.e., $1/\omega$, determines the wave length $1/\omega = \lambda/(2\pi)$ in space.) We realize from the formulas for $\bar{u}(\bar{x},0)$ and $\bar{u}(0,\bar{t})$ that there are three key dimensionless parameters related to these specific choices of initial and boundary conditions:

$$\alpha = \frac{A}{a}, \quad \beta = \frac{L}{\sigma}, \quad \gamma = kL = 2\pi\frac{L}{\lambda}.$$

With α, β, and γ we can write the dimensionless initial and boundary conditions as

$$\bar{u}(\bar{x},0) = \alpha e^{-\beta^2(\bar{x}-\frac{1}{2})^2},$$
$$\bar{u}(0,\bar{t}) = \sin(\gamma\bar{t}).$$

The dimensionless parameters have the following interpretations:

- α: ratio of initial condition peak and amplitude of incoming wave at $x = 0$

- β: ratio of length of domain and width of initial condition
- γ: ratio of length of domain and wave length of incoming wave

Again, these dimensionless parameters tell a lot about the interplay of the physical effects in the problem: only some ratios count.

We can simulate two special cases: $\alpha = 10$ (large) where the incoming wave is small and the solution is dominated by the two waves arising from $I(x)$, and $\alpha = 0.1$ (small) where the incoming waves dominate and the has the initial condition as a small perturbation of the wave shape. We may choose a peak-shaped initial condition: $\beta = 10$, and also a relatively short incoming wave compared to the domain size: $\gamma 6\pi$ (i.e., wave length of incoming wave is $L/6$). The function below applies the general unscaled solver in `wave1D_dn.py`[18] for solving the wave equation with constant c and any time-dependent function or $\partial u/\partial x = 0$ at the end points.

```
def simulate_Gaussian_and_incoming_wave():
    from wave1D_dn import solver, viz
    from math import pi
    alpha = 0.1
    beta = 10
    gamma = 2*pi*3

    def I(x):
        return alpha*exp(-beta**2*(x - 0.5)**2)

    def U_0(t):
        return sin(gamma*t) if t <= 2*pi/gamma else 0

    L = 1
    c = 1
    Nx = 80; dx = L/float(Nx); dt = dx/1
    #solver(I=I, V=0, f=0, U_0=U_0, U_L=None, L=1, dt=dt, C=1, T=4,
    #       user_action=myplotter)
    viz(I=I, V=0, f=0, c=1, U_0=U_0, U_L=None, L=1, dt=dt, C=1,
        T=4, umin=-(alpha+1), umax=(alpha+1),
        version='vectorized', animate=True)
```

The function is found in the file `session.py`[19]. Figures 11 and 12 shows snapshots of how $\bar{u}(\bar{x}, \bar{t})$ evolves due to a large/small initial condition and small/large incoming wave at the left boundary.

4.4 Velocity initial condition

Now we change the initial condition from $u = I$ and $\partial u/\partial t = 0$ to

$$u(x, 0) = 0, \tag{110}$$

$$\frac{\partial}{\partial t} u(x, 0) = V(x)\,. \tag{111}$$

Impact problems are often of this kind. From (98) we now get $f_L + f_R = 0$ and $cf'_L - cf'_R = V$. Introducing $W(x)$ such that $W'(x) = V(x)$, a solution is $-f_L = \frac{1}{2}W$ and $f_R = \frac{1}{2}W$. Hence,

$$u(x, t) = \frac{1}{2c} \int_{x-ct}^{x+ct} v(\xi) d\xi\,.$$

[18] `http://tinyurl.com/nm5587k/wave/wave1D/wave1D_dn.py`
[19] `http://tinyurl.com/nm5587k/scale/session.py`

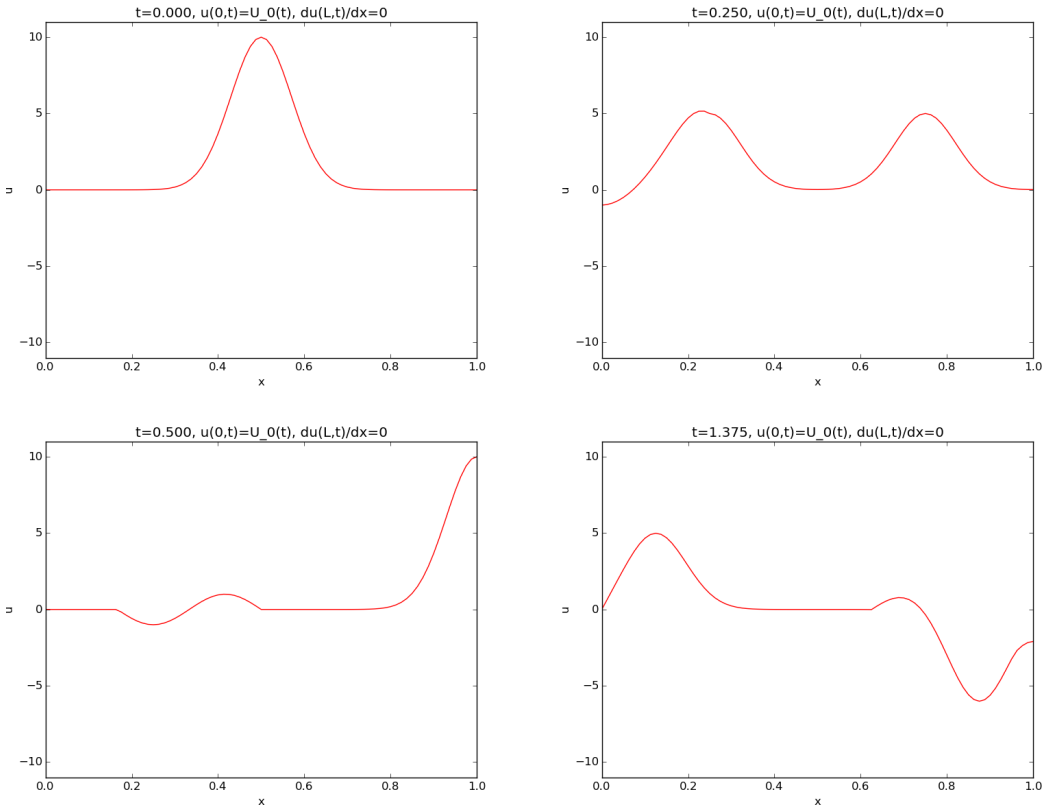

Figure 11: Snapshots of solution with large initial condition and small incoming wave ($\alpha = 10$).

The scaled version of $u_t(x,0) = V(x)$ becomes

$$\frac{\partial}{\partial \bar{t}}\bar{u}(\bar{x},0) = \frac{t_c}{u_c}V(\bar{x}x_c)\,.$$

Since V is the time-derivative of u, the characteristic size of V is typically u_c/t_c, meaning that

$$\max_{x\in(0,L)} |V(x)| = \frac{u_c}{t_c},$$

which gives $u_c = \max_{x\in(0,L)} |V(x)|L/c$. As usual, we base t_c on the wave speed: $t_c = L/c$. We end up with

$$\frac{\partial}{\partial \bar{t}}\bar{u}(\bar{x},0) = \frac{V(\bar{x}x_c)}{\max_x |V(x)|},$$

which also looks like a natural scaling of a function V.

Suppose we change the initial condition $u(x,0) = 0$ to $u(x,0) = I(x)$. The scaled version of this condition with the present u_c becomes

$$\bar{u}(\bar{x},0) = \frac{cI(\bar{x}x_c)}{L\max_x |V(x)|}\,.$$

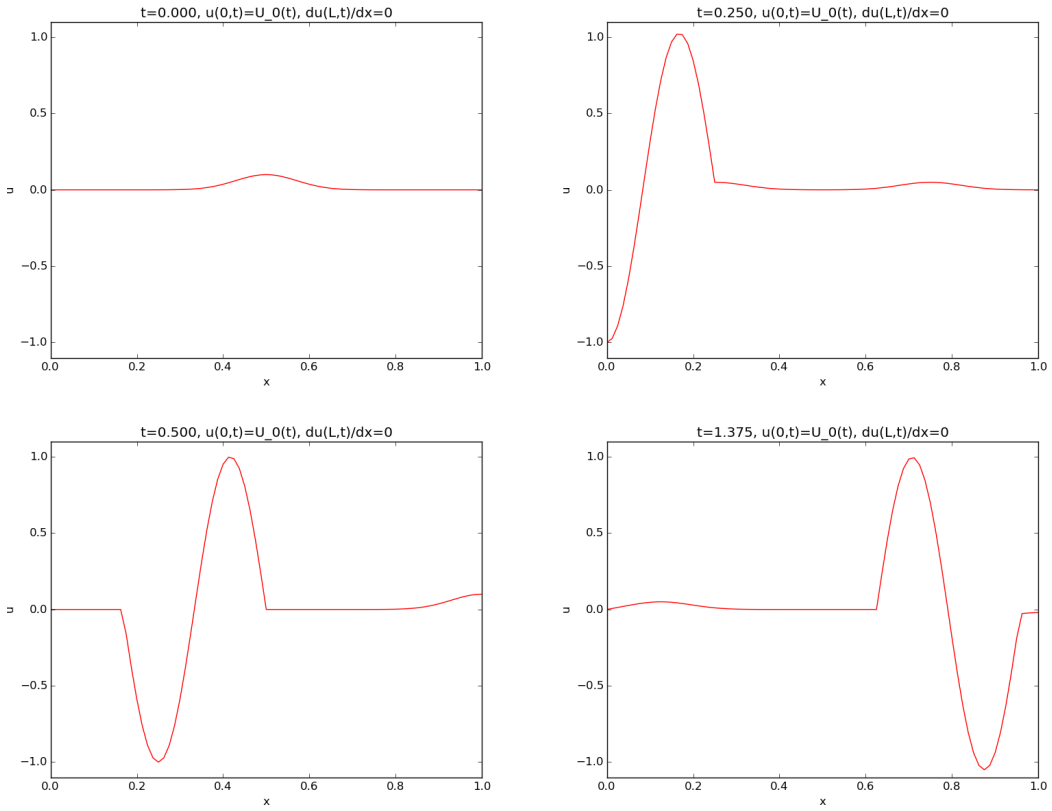

Figure 12: Snapshots of solution with small initial condition and large incoming wave ($\alpha = 0.1$).

Check that dimensionless numbers are dimensionless!

Is the fraction on the right-hand side dimensionless? It is easy to make errors when scaling equations, so checking that such fractions are dimensionless is wise. The dimension of I is the same as u, here taken to be displacement: [L]. Since V is $\partial u/\partial t$, its dimension is $[\mathrm{LT}^{-1}]$. The dimensions of c and L are $[\mathrm{LT}^{-1}]$ and [L]. The dimension of the right-hand side is then

$$\frac{[\mathrm{LT}^{-1}][L]}{[L][L\mathrm{T}^{-1}]} = 1,$$

demonstrating that the fraction is indeed dimensionless.

One may introduce a dimensionless initial shape, $\bar{I}(\bar{x}) = I(\bar{x}L)/\max_x |I|$. Then

$$\bar{u}(\bar{x}, 0) = \alpha \bar{I}(\bar{x}),$$

where α the dimensionless number

$$\alpha = \frac{c}{L}\frac{\max_x |I(x)|}{\max_x |V(x)|}.$$

If V is much larger than I, one expects that the influence of I is small. However, it takes time for the initial velocity V to influence the wave motion, so if c is much bigger than L, the initial wave shape I travels quickly through the domain before the effect of V becomes visible. The impact of I may therefore be significant for small t. This is reflected in an α value that is not small since c/L is large and $\max|I|/\max|V|$ is small, resulting in a scaled initial condition $\bar{u}(\bar{x},0)$ that is not small. With c/L about unity, α becomes small, and $\bar{u}(\bar{x},0)\approx 0$ such that not much happens before the effect of V becomes visible. Recall that the dimensionless initial velocity is about unity regardless of other parameters. Again, the scaling and the resulting dimensionless parameter(s) teach us much about the interaction of the various physical effects.

hpl 4: Do experiments. Make exercise or insert here.

hpl 5: Could make paradox: small I, big V, but still significant impact of I in a simulation. Why? Bug? The answer is above. Best as exercise.

4.5 Variable wave velocity and forcing

The next problem generalization regards wave propagation in a non-homogeneous where the wave velocity c depends on the spatial position: $c = c(x)$. To simplify the notation we introduce $\lambda(x) = c^2(x)$. We introduce homogeneous Neumann conditions at $x = 0$ and $x = L$. In addition, we add a force term $f(x,t)$ to the PDE, modeling wave generation in the interior of the domain. For example, a moving slide at the bottom of a fjord will generate surface waves and is modeled by such an $f(x,t)$ term. The initial-boundary value problem can be then expressed as

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(\lambda(x)\frac{\partial u}{\partial x}\right) + f(x,t), \qquad x\in(0,L),\ t\in(0,T], \tag{112}$$

$$u(x,0) = I(x), \qquad x\in[0,L], \tag{113}$$

$$\frac{\partial}{\partial t}u(x,0) = 0, \qquad x\in[0,L], \tag{114}$$

$$\frac{\partial}{\partial x}u(0,t) = 0, \qquad t\in(0,T], \tag{115}$$

$$\frac{\partial}{\partial x}u(L,t) = 0, \qquad t\in(0,T]. \tag{116}$$

We make the coefficient λ non-dimensional by

$$\bar{\lambda}(\bar{x}) = \frac{\lambda(\bar{x}x_c)}{\lambda_c}, \tag{117}$$

where one normally chooses the characteristic size of λ, λ_c, to be the maximum value such that $|\lambda|\leq 1$:

$$\lambda_c = \max_{x\in(0,L)} \lambda(x)\,.$$

Similarly, f has a scaled version

$$\bar{f}(\bar{x},\bar{t}) = \frac{f(\bar{x}x_c,\bar{t}t_c)}{f_c},$$

where normally

$$f_c = \max_{x,t}|f(x,t)|\,.$$

Inserting dependent and independent variables expressed by their non-dimensional counterparts yields

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{t_c^2 \lambda_c}{L^2} \frac{\partial}{\partial \bar{x}} \left(\bar{\lambda}(\bar{x}) \frac{\partial \bar{u}}{\partial \bar{x}} \right) + \frac{t_c^2 f_c}{u_c} \bar{f}(\bar{x}, \bar{t}), \qquad \bar{x} \in (0,1),\ \bar{t} \in (0, \bar{T}], \tag{118}$$

$$\bar{u}(\bar{x}, 0) = \frac{I(x)}{u_c}, \qquad \bar{x} \in [0,1], \tag{119}$$

$$\frac{\partial}{\partial \bar{t}} \bar{u}(\bar{x}, 0) = 0, \qquad \bar{x} \in [0,1], \tag{120}$$

$$\frac{\partial}{\partial \bar{x}} \bar{u}(0, \bar{t}) = 0, \qquad \bar{t} \in (0, \bar{T}], \tag{121}$$

$$\frac{\partial}{\partial \bar{x}} \bar{u}(1, \bar{t}) = 0, \qquad \bar{t} \in (0, \bar{T}], \tag{122}$$

with $\bar{T} = Tc/L$.

The time scale is, as before, chosen as $t_c = L/\sqrt{\lambda_c}$. Note that the previous (constant) wave velocity c now corresponds to $\sqrt{\lambda(x)}$. Therefore, $\sqrt{\lambda_c}$ is a characteristic wave velocity.

One could wonder if the time scale of the force term, $f(x,t)$, should influence t_c, but as we reasoned for the boundary condition $u(0,t) = U_L(t)$, we let the characteristic time be governed by the signal speed in the medium, i.e., by $\sqrt{\lambda_c}$ here and not by the time scale of the excitation f which dictates the length of the generated waves and not their propagation speed.

Furthermore, we may choose u_c as $\max_x |I(x)|$, as before, or we may fit u_c such that the coefficient in the source term is unity, i.e., all terms balance each other. This latter idea leads to

$$u_c = \frac{L^2 f_c}{\lambda_c}$$

and a PDE without parameters,

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{\partial}{\partial \bar{x}} \left(\bar{\lambda}(\bar{x}) \frac{\partial \bar{u}}{\partial \bar{x}} \right) + \bar{f}(\bar{x}, \bar{t}) \,.$$

The initial condition $u(x,0) = I(x)$ becomes in dimensionless form

$$\bar{u}(\bar{x}, 0) = u_c^{-1} \max_x |I(x)| \bar{I}(\bar{x}) = \beta^{-1} \bar{I}(\bar{x}),$$

where

$$\beta = \frac{L^2}{\lambda_c} \frac{\max_{x,t} |f(x,t)|}{\max_x |I(x)|} \,.$$

In the case $u_c = \max_x |I(x)|$, $\bar{u}(\bar{x}, 0) = \bar{I}(\bar{x})$ and the β parameter appears in the PDE,

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{\partial}{\partial \bar{x}} \left(\bar{\lambda}(\bar{x}) \frac{\partial \bar{u}}{\partial \bar{x}} \right) + \beta \bar{f}(\bar{x}, \bar{t}) \,.$$

With $V = 0$, and $u = 0$ or $u_x = 0$ on the boundaries $x = 0, L$, this scaling gives $|\bar{u}| \leq 1$, since initially $|I| \leq 1$, and no boundary condition can increase the amplitude.

The initial condition $u_t(x,0) = V(x)$ has its dimensionless variant as

$$\bar{V}(\bar{x}) = \frac{t_c}{u_c} \frac{V(L\bar{x})}{\max_x |V(x)|},$$

which becomes

$$\frac{\partial \bar{u}}{\partial \bar{t}}(\bar{x},0) = \frac{L}{\sqrt{\lambda_c}}\frac{\max_x |V(x)|}{\max_x |I(x)|}\bar{V}(\bar{x}),\ \text{if } u_c = \max_x |I(x)|,$$

or

$$\frac{\partial \bar{u}}{\partial \bar{t}}(\bar{x},0) = \frac{\sqrt{\lambda_c}}{L}\frac{\max_x |V(x)|}{\max_{x,t} |f(x,t)|}\bar{V}(\bar{x}),\ \text{if } u_c = t_c^2 f_c = \frac{L^2}{\lambda_c}\max_{x,t}|f(x,t)|\,.$$

Introducing the dimensionless number α (cf. Section 4.4),

$$\alpha^{-1} = \frac{\sqrt{\lambda_c}}{L}\frac{\max_x |V(x)|}{\max_{x,t} |f(x,t)|},$$

we can write

$$\frac{\partial \bar{u}}{\partial \bar{t}}(\bar{x},0) = \begin{cases} \alpha^{-1}\bar{V}(\bar{x}), & u_c = \max_x |I|, \\ \alpha^{-1}\beta^{-1}\bar{V}(\bar{x}), & u_c = t_c^2 f_c \end{cases}$$

4.6 Damped wave equation

A linear damping term $b\,\partial u/\partial t$ is often added to the wave equation to model energy dissipation and amplitude reduction.

$$\frac{\partial^2 u}{\partial t^2} + b\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\lambda(x)\frac{\partial u}{\partial x}\right) + f(x,t)\,. \tag{123}$$

The scaled equation becomes

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} + \frac{t_c}{b}\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{t_c^2\lambda_c}{L^2}\frac{\partial}{\partial \bar{x}}\left(\bar{\lambda}(\bar{x})\frac{\partial \bar{u}}{\partial \bar{x}}\right) + \frac{t_c^2 f_c}{u_c}\bar{f}(\bar{x},\bar{t})\,.$$

The damping term is usually much smaller than the two other terms involving $\bar{u}$. The time scale is therefore chosen as in the undamped case, $t_c = L/\sqrt{\lambda_c}$. As in Section 4.5, we have two choices of u_c: $u_c = \max_x |I|$ or $u_c = t_c^2 f_c$. The former choice of u_c gives a PDE with two dimensionless numbers,

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} + \gamma\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial}{\partial \bar{x}}\left(\bar{\lambda}(\bar{x})\frac{\partial \bar{u}}{\partial \bar{x}}\right) + \beta\bar{f}(\bar{x},\bar{t}), \tag{124}$$

where

$$\gamma = \frac{bL}{\sqrt{\lambda_c}},$$

measures the size of the damping. With $u_c = t_c^2 f_c$ we get a PDE where only γ enters,

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} + \gamma\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial}{\partial \bar{x}}\left(\bar{\lambda}(\bar{x})\frac{\partial \bar{u}}{\partial \bar{x}}\right) + \bar{f}(\bar{x},\bar{t})\,. \tag{125}$$

The scaled initial conditions are as in Section 4.5.

To summarize, the effects of V, f, and damping are reflected in the dimensionless numbers α, β, and γ, respectively.

4.7 A three-dimensional wave equation problem

To demonstrate how the scaling extends to and looks like in three-dimensions, we consider

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{\partial}{\partial x}\left(\lambda \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda \frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda \frac{\partial u}{\partial z}\right). \tag{126}$$

We introduce

$$\bar{x} = \frac{x}{x_c}, \quad \bar{y} = \frac{y}{y_c}, \quad \bar{z} = \frac{z}{z_c}, \quad \bar{t} = \frac{t}{t_c}, \quad \bar{u} = \frac{u}{u_c}.$$

With $\bar{\lambda} = \lambda(\bar{x}x_c, \bar{y}y_c, \bar{z}z_c)/\lambda_c$, we get

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{t_c^2\lambda_c}{x_c^2}\frac{\partial}{\partial \bar{x}}\left(\bar{\lambda}\frac{\partial \bar{u}}{\partial \bar{x}}\right) + \frac{t_c^2\lambda_c}{y_c^2}\frac{\partial}{\partial \bar{y}}\left(\bar{\lambda}\frac{\partial \bar{u}}{\partial \bar{y}}\right) + \frac{t_c^2\lambda_c}{z_c^2}\frac{\partial}{\partial \bar{z}}\left(\bar{\lambda}\frac{\partial \bar{u}}{\partial \bar{z}}\right).$$

Often, we will set $x_c = y_c = z_c = L$ where L is some characteristic size of the domain. As before, $t_c = L/\sqrt{\lambda_c}$, and these choices lead to a dimensionless wave equation without physical parameters:

$$\frac{\partial^2 \bar{u}}{\partial \bar{t}^2} = \frac{\partial}{\partial \bar{x}}\left(\bar{\lambda}\frac{\partial \bar{u}}{\partial \bar{x}}\right) + \frac{\partial}{\partial \bar{y}}\left(\bar{\lambda}\frac{\partial \bar{u}}{\partial \bar{y}}\right) + \frac{\partial}{\partial \bar{z}}\left(\bar{\lambda}\frac{\partial \bar{u}}{\partial \bar{z}}\right). \tag{127}$$

The initial conditions remain the same as in the previous one-dimensional examples.

5 The diffusion equation

The diffusion equation in a one-dimensional homogeneous medium reads

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \quad x \in (0, L),\ t \in (0, T], \tag{128}$$

where α is the diffusion coefficient. The multi-dimensional generalization to a heterogeneous medium and a force term takes the form

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha \nabla u) + f, \quad x, y, z \in \Omega,\ t \in (0, T]. \tag{129}$$

We first look at scaling the PDE itself, and thereafter we discuss some types of boundary conditions and how to scale the complete initial-boundary value problem.

5.1 Homogeneous diffusion equation

Simplified 1D PDE. To make (128) dimensionless, we introduce as usual dimensionless dependent and independent variables:

$$\bar{x} = \frac{x}{x_c}, \quad \bar{t} = \frac{t}{t_c}, \quad \bar{u} = \frac{u}{u_c}.$$

Inserting the dimensionless quantities in the one-dimensional PDE (128) results in

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{t_c \alpha}{L^2}\frac{\partial^2 \bar{u}}{\partial \bar{x}^2}, \quad \bar{x} \in (0, 1),\ \bar{t} \in (0, \bar{T} = T/t_c].$$

Arguing as for the wave equation that the scaling should result in

$$\frac{\partial \bar{u}}{\partial \bar{t}} \text{ and } \frac{\partial^2 \bar{u}}{\partial \bar{x}^2}$$

of the same size (about unity), implies $t_c\alpha/L^2 = 1$ and therefore $t_c = L^2/\alpha$.

Insight through an analytical solution can alternatively help with choosing t_c. One can show that $u = Ae^{-pt}\sin(kx)$ is a solution of (5.1) if $p = \alpha k^2$, for any k. Exponential decay in time is a characteristic feature of the diffusion equation, and the e-folding time can then be taken as a time scale. This means $t_c = 1/p \sim k^{-2}$. Since k is related to the spatial wave length λ through $k = 2\pi/\lambda$, it means that t_c depends strongly on the wave length of the sine term $\sin(kx)$. In particular, short waves (as found in noisy signals) with large k decay very rapidly. For the overall solution we are interested in how the longest meaningful wave decays and use that time scale for t_c. The longest wave typically has half a wave length over the domain $[0, L]$: $u = Ae^{-pt}\sin(\pi x/L)$ ($k = \pi/L$). Then $t_c = L^2/\alpha\pi^{-2}$, but factor π^{-2} is not important and we simply choose $t_c = L^2/\alpha$, which equals the time scale we arrived at above. We may say that t_c is the time it takes for the diffusion to significantly change the solution in the entire domain.

Another fundamental solution of the diffusion equation is the diffusion of a Gaussian function: $u(x,t) = (4\pi\alpha t)^{-1/2}\exp(-x^2/(4\alpha t))$. For the diffusion to be significant at a distance $x = L$, we may demand the exponential factor to have a value of $e^{-1} \approx 0.37$, which implies $t = L^2/(4\alpha)$, but the factor 4 is not of importance, so again, a relevant time scale is $t_c = L^2/\alpha$.

The scale u_c is chosen according to the initial condition: $u_c = \max_{x\in(0,L)}|I(x)|$. For a diffusion equation $u_t = \alpha u_{xx}$ with $u = 0$ at the boundaries $x = 0, L$, the solution is bounded by the initial condition $I(x)$. Therefore, the listed choice of u_c implies that $|u| \leq 1$. (The solution $u = Ae^{-pt}\sin(kx)$ is such an example if $k = n\pi/L$ for integer n such that $u = 0$ for $x = 0$ and $x = L$.)

The resulting dimensionless PDE becomes

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial^2 \bar{u}}{\partial \bar{x}^2}, \quad \bar{x} \in (0,1),\ \bar{t} \in (0, \bar{T}], \tag{130}$$

with initial condition

$$\bar{u}(\bar{x}, 0) = \bar{I}(\bar{x}) = \frac{I(x_c\bar{x})}{\max_x |I(x)|}\,.$$

Notice that (130) is without physical parameters!

Generalized PDE. Turning the attention to (129), we introduce the dimensionless diffusion coefficient

$$\bar{\alpha}(\bar{x}, \bar{y}, \bar{z}) = \alpha_c^{-1}\alpha(x_c\bar{x}, y_c\bar{y}, z_c\bar{z}),$$

typically with

$$\alpha_c = \max_{x,y,z} \alpha(x, y, z)\,.$$

The length scales are

$$\bar{x} = \frac{x}{x_c}, \quad \bar{y} = \frac{y}{y_c}, \quad \bar{z} = \frac{z}{z_c}\,.$$

We scale f in a similar fashion:

$$\bar{f}(\bar{x}, \bar{y}, \bar{z}, \bar{t}) = f_c^{-1} f(\bar{x}x_c, \bar{y}y_c\bar{z}z_c, \bar{t}t_c),$$

with

$$f_c = \max_{x,y,z,t} |f(x,y,z,t)|\,.$$

Also assuming that $x_c = y_c = z_c = L$, and $u_c = \max_{x,y,z}(I(x,y,z)$, we end up with the scaled PDE

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \bar{\nabla}\cdot\left(\bar{\alpha}\bar{\nabla}\bar{u}\right) + \beta f, \quad \bar{x},\bar{y},\bar{z} \in \bar{\Omega},\ \bar{t} \in (0,\bar{T}]\,. \tag{131}$$

Here, $\bar{\nabla}$ means differentiation with respect to dimensionless coordinates $\bar{x}$, $\bar{y}$, and $\bar{z}$. The dimensionless parameter β takes the form

$$\beta = \frac{t_c f_c}{u_c} = \frac{L^2}{\alpha}\frac{\max_{x,y,z,t}|f(x,y,z,t)|}{\max_{x,y,z}|I(x,y,z)|}\,.$$

The scaled initial condition is $\bar{u} = \bar{I}$ as in the 1D case.

An alternative choice u_c is to make the coefficient $t_c f_c/u_c$ in the source term unity. The scaled PDE now becomes

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \bar{\nabla}\cdot\left(\bar{\alpha}\bar{\nabla}\bar{u}\right) + f, \tag{132}$$

but the initial condition features the β parameter:

$$\bar{u}(\bar{x},\bar{y},\bar{z},0) = \frac{I}{t_c f_c} = \beta^{-1}\bar{I}(\bar{x},\bar{y},\bar{z})\,.$$

We may check that β is really non-dimensional. From the PDE, f must have the same dimensions as $\partial u/\partial t$, i.e., $[\Theta \mathrm{T}^{-1}]$. The dimension of α is more intricate, but from the term αu_{xx} we know that u_{xx} has dimensions $[\Theta \mathrm{L}^{-2}]$, and then α must have dimension $\mathrm{L}^2\mathrm{T}^{-1}]$ to match the target $[\Theta \mathrm{T}^{-1}]$. In the expression for β we get $[\mathrm{L}^2\Theta\mathrm{T}^{-1}(\mathrm{L}^2\mathrm{T}^{-1}\Theta)^{-1}]$, which equals 1 as it should.

5.2 Jump boundary condition

A classical one-dimensional heat conduction problem goes as follows. An insulated rod at some constant temperature U_0 is suddenly heated from one end ($x = 0$), modeled as a constant Dirichlet condition $u(0,t) = U_L \neq U_0$ at that end. That is, the boundary temperature jumps from U_0 to U_1 at $t = 0$. All the other surfaces of the rod are insulated such that a one-dimensional model is appropriate. Heat cannot escape, and we supply heat at $x = 0$ at the temperature U_1, which will warm up all of the material to the temperature U_1. That is, $u \to U_1$ as $t \to \infty$.

The initial-boundary value problem reads

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2}, \qquad x \in (0,L),\ t \in (0,T], \tag{133}$$

$$u(x,0) = U_0, \qquad x \in [0,L], \tag{134}$$

$$u(0,t) = U_1, \qquad t \in (0,T], \tag{135}$$

$$\frac{\partial}{\partial x}u(L,t) = 0, \qquad t \in (0,T]. \tag{136}$$

The diffusion coefficient is related to heat transfer parameters by $\alpha = k/(\varrho c)$, where k is the heat conduction coefficient, ϱ is the density, and c is a specific heat capacity parameter.

The natural dimensionless temperature for this problem is

$$\bar{u} = \frac{u - U_0}{U_1 - U_0},$$

since this choice makes $\bar{u} \in [0,1]$. The reason is that u is bounded by the initial and boundary conditions (in the absence of a source term), and we have $\bar{u}(\bar{x},0) = 0$, $\bar{u}(\bar{x},\infty) = 1$, and $\bar{u}(0,\bar{t}) = 1$.

The choice of t_c is as in the previous cases. We arrive at the dimensionless initial-boundary value problem

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial^2 \bar{u}}{\partial \bar{x}^2}, \qquad \bar{x} \in (0,1),\ \bar{t} \in (0,\bar{T}], \tag{137}$$

$$\bar{u}(\bar{x},0) = 0, \qquad \bar{x} \in [0,1], \tag{138}$$

$$\bar{u}(0,\bar{t}) = 1, \qquad \bar{t} \in (0,\bar{T}], \tag{139}$$

$$\frac{\partial}{\partial \bar{x}} u(1,\bar{t}) = 0, \qquad \bar{t} \in (0,\bar{T}]. \tag{140}$$

The striking feature is that there are no physical parameters left in this problem. One simulation can be carried out for $\bar{u}(\bar{x},\bar{t})$, and the temperature in a rod of any material and any constant initial and boundary temperature can be retrieved by

$$u(x,t) = U_0 + (U_1 - U_0)\bar{u}(x/L, t\alpha/L^2)\,.$$

hpl 6: Include simulation results.

5.3 Oscillating Dirichlet condition

Let us address the heat equation problem where the temperature is oscillating on the boundary $x = 0$:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad x \in (0,L),\ t \in (0,T], \tag{141}$$

$$u(x,0) = U_0, \qquad x \in [0,L], \tag{142}$$

$$u(0,t) = U_0 + A\sin(\omega t), \qquad t \in (0,T], \tag{143}$$

$$\frac{\partial}{\partial x} u(L,t) = 0, \qquad t \in (0,T]. \tag{144}$$

One important physical application is temperature oscillations in the ground, either day and night variations at a short temporal and spatial scale, or seasonal variations in the Earth's crust.

Since the boundary temperature is oscillating around the initial condition, we expect $u \in [-A, A]$. Then the dimensionless temperature is therefore taken as

$$\bar{u} = \frac{u - U_0}{2A},$$

such that $\bar{u} \in [-1,1]$.

What is an appropriate time scale? There will be two time scales involved, the oscillations $\sin(\omega t)$ with period $P = 2\pi/\omega$ at the boundary and the "speed of diffusion" where $t_c = L^2/\alpha$ is the appropriate scale. Analytical insight exists in this model problem because the exact solution is known to be

$$u(x,t) = U_0 - Ae^{-bx}\sin(bx - \omega t), \quad b = \sqrt{\frac{\omega}{2\alpha}}\,. \tag{145}$$

This solution is of the form $e^{-bx}g(x-ct)$, i.e., a wave that moves to the right with velocity c and a damped amplitude e^{-bx}. This is seen if we make a rewrite

$$u(x,t) = U_0 - Ae^{-bx}\sin\left(b(x-ct)\right), \quad c = \omega/b = \sqrt{2\alpha\omega},\ b = \sqrt{\frac{\omega}{2\alpha}}\,.$$

The boundary oscillations lead to the time scale $t_c = 1/\omega$. The speed of the wave suggests another time scale: the time it takes to propagate through the domain, which is L/c, and hence $t_c = L/c = L/\sqrt{2\alpha\omega}$.

One may argue that L is not the appropriate length scale, because u is damped by e^{-bx} so for $x > 4/b$, u is close to zero. Using $1/b$ as length scale, which is the e-folding distance of the damping factor, and basing t_c on the time it takes a signal to propagate one length scale, $t_c^{-1} = bc = \omega$. Similarly, the time scale based on the "speed of diffusion" changes to $t_c^{-1} = b^2\alpha = \frac{1}{2}\omega$ if we employ $1/b$ as length scale.

To summarize, we have three candidates for the time scale: $t_c = L^2/\alpha$, $t_c = 2/\omega$, and $t_c = 1/\omega$.

Let us look at the dimensionless exact solution to see if it can help with the choice. We introduce the dimensionless parameters

$$\beta = bx_c = x_c\sqrt{\frac{\omega}{2dfc}}, \quad \gamma = \omega t_c\,.$$

The scaled solution becomes

$$\bar{u}(\bar{x},\bar{t};\beta,\gamma) = e^{-\beta\bar{x}}\sin(\beta\bar{x} - \gamma\bar{t})\,.$$

The three choices of γ, implied by the three choices of t_c, are

$$\gamma = \begin{cases} 1, & t_c = 1/\omega, \\ 2, & t_c = 2/\omega, \\ 2\beta^2, & t_c = L^2/\alpha,\ x_c = L \end{cases} \tag{146}$$

The former two choices leaves only β as parameter in $\bar{u}$, and with $x_c = 1/b$ as length scale, β becomes unity, and there are no parameters in the dimensionless solution:

$$\bar{u}(\bar{x},\bar{t}) = e^{-\bar{x}}\sin(\bar{x} - \bar{t})\,. \tag{147}$$

Therefore, $x_c = 1/b$ and $t_c = 1/\omega$ (or $t_c = 2/\omega$, but the factor 2 is of no importance) are therefore the most appropriate scales.

To further argue why (147) these scales are preferred, think of ω as large. Then the wave is damped over a short distance and there will be a thin boundary layer of temperature oscillations near $x = 0$ and little changes in u in the rest of the domain. The scaling (147) resolves this problem by using $1/b \sim \omega^{-1/2}$ as length scale, because then the boundary layer thickness is independent of ω. The length of the domain can be chosen as, e.g., $4/b$ such that $\bar{u} \approx 0$ at this end. The length scale $1/b$ help us to zoom in on the part of u where significant changes take place.

In the other limit, ω small, b becomes small, and the wave is hardly damped in the domain $[0, L]$ unless L is large enough. The imposed boundary condition on $x = L$ in fact requires u to be constant so its derivative vanishes, and this property can only be obtained if L is enough to

ensure that the wave becomes significantly damped. Therefore, the length scale is dictated by b, not L, and L should be adapted to b, typically $L \geq 4/b$.

The resulting scaled problem becomes

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{1}{2}\frac{\partial^2 u}{\partial x^2}, \qquad \bar{x} \in (0, 4/b),\ \bar{t} \in (0, \bar{T}], \tag{148}$$

$$\bar{u}(\bar{x}, 0) = 0, \qquad \bar{x} \in [0, 1] \tag{149}$$

$$\bar{u}(0, \bar{t}) = \sin(\bar{t}), \qquad \bar{t} \in (0, \bar{T}], \tag{150}$$

$$\frac{\partial}{\partial \bar{x}}\bar{u}(1, \bar{t}) = 0, \qquad \bar{t} \in (0, \bar{T}]. \tag{151}$$

The coefficient in front of the second-derivative becomes

$$\frac{t_c \alpha}{1/b^2} = \frac{b^2 \alpha}{\omega} == \frac{1}{2}\,.$$

We may, of course, choose $t_c = 2/\omega$ and get rid of the $\frac{1}{2}$ factor, if desired, but then it turns up in (150) instead, as $\sin(2\bar{t})$.

5.4 Diffusion equation with source term

Let us add a source term $f(x,t)$ to the diffusion equation,

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + f(x,t), \quad x \in (0, L),\ t \in (0, T]\,. \tag{152}$$

As always, we introduce

$$\bar{x} = \frac{x}{x_c}, \quad \bar{t} = \frac{t}{t_c}, \quad \bar{u} = \frac{u}{u_c},$$

and a scaling of f,

$$\bar{f}(\bar{x}, \bar{t}) = \frac{f(x,t)}{f_c}, \quad f_c = \max_{x,t} f(x,t)\,.$$

Inserting the dimensionless quantities in (152) and making each term dimensionless results in

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{t_c \alpha}{x_c^2}\frac{\partial^2 u}{\partial x^2} + \frac{t_c f_c}{u_c}\bar{f}(\bar{x}, \bar{t}), \quad \bar{x} \in (0, L/x_c),\ \bar{t} \in (0, T/t_c]\,. \tag{153}$$

The magnitude of u is governed by f as well as by initial and boundary conditions. Suppose initial and boundary conditions are zero. The length scale x_c is naturally chosen as L such that $\bar{x} \in [0,1]$. It is also natural to choose t_c to be the time scale of diffusion, as before, $t_c = x_c^2/\alpha = L^2/\alpha$. The source term can match the unit size of the two other terms by choosing its coefficient to be unity, i.e.,

$$u_c = t_c f_c = \frac{L^2 \max_{x,t} f}{\alpha}\,.$$

The scaled equation then becomes free of physical parameters:

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial^2 u}{\partial x^2} + \bar{f}(\bar{x}, \bar{t}), \quad \bar{x} \in (0, L/x_c),\ \bar{t} \in (0, T/t_c]\,. \tag{154}$$

With an initial condition $u = I$, we get a scaled initial condition

$$\bar{u}(\bar{x}, 0) = \beta \bar{I}(L\bar{x}),$$

where $\bar{I} = I/\max_x I(x)$ is a scaled version of I, and β is a dimensionless number

$$\beta = \frac{\max_x I(x)}{\max_{x,t} f(x,t)} \frac{\alpha}{L^2}\,.$$

It measures the ratios of I and f per unit characteristic time.

Suppose now that we choose $u_c = \max_x I(x)$ because f is small compared to I. The initial condition then reads $\bar{u}(\bar{x}, 0) = \bar{I}(\bar{x})$, and the differential equation becomes

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial^2 u}{\partial x^2} + \beta^{-1} \bar{f}(\bar{x}, \bar{t})\,.$$

This is the relevant scaling for a small f and large β, indicating that $\beta^{-1}\bar{f}$ can be neglected from the PDE. A major influence of f leads to a balance between all three terms in the PDE, but the initial condition has a small β, and one can set $\bar{u}(\bar{x}, 0) \approx 0$. Exercise 16 explores a similar stationary problem.

5.5 Fisher's equation

Fisher's equation is essentially the logistic equation at each point for population dynamics combined with spatial movement through ordinary diffusion:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + \varrho u(1 - u/M)\,. \tag{155}$$

This PDE is also known as the KPP equation after Kolmogorov, Petrovsky, and Piskynov (who introduced the equation independently of Fisher).

Introducing

$$\bar{x} = \frac{x}{x_c}, \quad \bar{t} = \frac{t}{t_c}, \quad \bar{u} = \frac{u}{u_c},$$

results in

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{t_c \alpha}{x_c^2} \frac{\partial^2 u}{\partial x^2} + t_c \varrho \bar{u}(1 - u_c \bar{u}/M)\,.$$

If all terms are equally important, the scales can be determined from demanding the coefficients to be unity and by scaling u by M:

$$u_c = M, \quad t_c = \frac{1}{\varrho}, \quad x_c = \sqrt{\frac{\alpha}{\varrho}}\,.$$

The scaled PDE becomes

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial^2 u}{\partial x^2} + \bar{u}(1 - \bar{u})\,. \tag{156}$$

With this scaling, the length scale $x_c = \sqrt{\alpha/\varrho}$ is not related to the domain size, so the scale is most relevant for an infinite domain. Also, note that the time scale $t_c = \varrho^{-1}$ is governed by the logistic term (exponential growth for small u) and not the diffusion process as in most previous examples.

On a finite domain $[0, L]$ we set $x_c = L$, $t_c = \varrho^{-1}$, $u_c = M$, which gives

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \beta \frac{\partial^2 u}{\partial x^2} + \bar{u}(1 - \bar{u}), \tag{157}$$

where β is a dimensionless number

$$\beta = \frac{\alpha}{\varrho L^2} = \frac{\varrho^{-1}}{L^2/\alpha},$$

demonstrating that β measures the ratio of the time scale for exponential growth in the beginning of the logistic process and the time scale of diffusion L^2/α. For small β we can neglect the diffusion and spatial movements, and the PDE is essentially a logistic ODE at each point, while for large β, diffusion dominates, and t_c should in that case be based on the diffusion time scale L^2/α. This leads to the scaled PDE

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \frac{\partial^2 u}{\partial x^2} + \beta^{-1} \bar{u}(1 - \bar{u}), \tag{158}$$

showing that a large β encourages omission of the logistic term, because the point-wise growth takes place on large time scales while diffusion is rapid. The effect of diffusion is then more prominent and it suffices to solve $\bar{u}_{\bar{t}} = \bar{u}_{\bar{x}\bar{x}}$.

6 The convection-diffusion equation

6.1 Convection-diffusion without a force term

We now add a convection term $\boldsymbol{v} \cdot \nabla u$ to the diffusion equation to obtain the well-known convection-diffusion equation:

$$\frac{\partial u}{\partial t} + \boldsymbol{v} \cdot \nabla u = \alpha \nabla^2 u, \quad \bar{x}, \bar{y}, \bar{z} \in \bar{\Omega},\ t \in (0, T]\,. \tag{159}$$

The velocity field $\boldsymbol{v}$ is prescribed, and its characteristic size V is normally clear from the problem description.

hpl 7: Show a couple of cases, can be the same as Navier-Stokes, where different V is obvious.

Inserting

$$\bar{x} = \frac{x}{x_c},\ \bar{y} = \frac{y}{y_c},\ \bar{z} = \frac{z}{z_c},\ \bar{t} = \frac{t}{t_c},\ \bar{\boldsymbol{v}} = \frac{\boldsymbol{v}}{V},\ \bar{u} = \frac{u}{u_c}$$

in (159) yields

$$\frac{u_c}{t_c}\frac{\partial \bar{u}}{\partial \bar{t}} + \frac{u_c V}{L} \bar{\boldsymbol{v}} \cdot \bar{\nabla} \bar{u} = \frac{\alpha u_c}{L^2} \bar{\nabla}^2 \bar{u}, \quad \bar{x}, \bar{y}, \bar{z} \in \Omega,\ \bar{t} \in (0, \bar{T}]\,.$$

For u_c we simply introduce the symbol U, which we may estimate from an initial condition. It is not critical here since it vanishes from the scaled equation anyway as long as there is no source term present. With some velocity measure V and length measure L, it is tempting to just let $t_c = L/V$. The alternative is to use the diffusion length scale $t_c = L^2/\alpha$. Very often in these kind of problems, the convection term $\boldsymbol{v} \cdot \nabla u$ dominates over the diffusion term $\alpha \nabla^2 u$, so the time scale for convection, which is typically L/V (the time it takes for the convection velocity to propagate a signal through the characteristic length), is most appropriate of the two. Only when

the diffusion term is very much larger than the convection term (corresponding to very small Peclet numbers, see below) we would apply $t_c = L^2/\alpha$. The non-dimensional form of the PDE with $t_c = L/V$ becomes

$$\frac{\partial \bar{u}}{\partial \bar{t}} + \bar{\boldsymbol{v}} \cdot \bar{\nabla} \bar{u} = \mathrm{Pe}^{-1} \bar{\nabla}^2 \bar{u}, \quad \bar{x}, \bar{y}, \bar{z} \in \Omega,\ \bar{t} \in (0, \bar{T}], \tag{160}$$

where Pe is the *Peclet number*,

$$\mathrm{Pe} = \frac{LV}{\alpha}\,.$$

Estimating the size of the convection term $\boldsymbol{v} \cdot \nabla u$ as VU/L and the diffusion term $\alpha \nabla^2 u$ as $\alpha U/L^2$, we see that the Peclet number measures the ratio of the convection and the diffusion terms:

$$\mathrm{Pe} = \frac{\text{convection}}{\text{diffusion}} = \frac{VU/L}{\alpha U/L^2} = \frac{LV}{\alpha}\,.$$

In case we use the diffusion time scale $t_c = L^2/\alpha$, we get the non-dimensional PDE

$$\frac{\partial \bar{u}}{\partial \bar{t}} + \mathrm{Pe}\, \bar{\boldsymbol{v}} \cdot \bar{\nabla} \bar{u} = \bar{\nabla}^2 \bar{u}, \quad \bar{x}, \bar{y}, \bar{z} \in \Omega,\ \bar{t} \in (0, \bar{T}]\,. \tag{161}$$

Discussion of scales and balance of terms in the PDE.

We see that (160) and (161) are not equivalent, and they are based on two different time scales. For moderate Peclet numbers around 1, all terms have the same size in (160), i.e., a size around unity. For large Peclet numbers, (160) expresses a balance between the time derivative term and the convection term, both of size unity, and then there is a very small term $\mathrm{Pe}^{-1}\bar{\nabla}^2\bar{u}$ term because Pe is large and $\bar{\nabla}^2\bar{u}$ should be of size unity. That the convection term dominates over the diffusion term is consistent with the time scale $t_c = L/V$ based on convection transport. In this case, we can neglect the diffusion term as Pe goes to infinity and work with a pure convection (or advection) equation

$$\frac{\partial \bar{u}}{\partial \bar{t}} + \bar{\boldsymbol{v}} \cdot \bar{\nabla} \bar{u} = 0\,.$$

For small Peclet numbers, $\mathrm{Pe}^{-1}\bar{\nabla}^2\bar{u}$ becomes very large and can only be balanced by two terms that are supposed to be unity of size. The time-derivative and/or the convection term must be much larger than unity, but that means that we use wrong scales, since right scales imply that $\partial\bar{u}/\partial\bar{t}$ and $\bar{v} \cdot \bar{\nabla}\bar{u}$ of order unity. Switching to a time scale based on diffusion as the dominating physical effect gives (161). For very small Peclet numbers this equation tells that the time-derivative balances the diffusion, while the convection $\bar{\boldsymbol{v}} \cdot \bar{\nabla}\bar{\boldsymbol{u}}$ is around unity, but multiplied by a very small coefficient Pe, so this term is negligible in the PDE. An approximate PDE for small Peclet numbers is therefore

$$\frac{\partial \bar{u}}{\partial \bar{t}} = \bar{\nabla}^2 \bar{u}\,.$$

Scaling can with the above type of reasoning be used to neglect terms from a differential equation under precise mathematical conditions.

6.2 Stationary PDE

Suppose the problem is stationary and that there is no need for any time scale. How is this type of convection-diffusion problem scaled? We get

$$\frac{VU}{L}\bar{\boldsymbol{v}}\cdot\bar{\nabla}\bar{u} = \frac{\alpha U}{L^2}\bar{\nabla}^2\bar{u},$$

or

$$\bar{\boldsymbol{v}}\cdot\bar{\nabla}\bar{u} = \mathrm{Pe}^{-1}\bar{\nabla}^2\bar{u}\,. \tag{162}$$

This scaling only "works" for moderate Peclet numbers. For very small or very large Pe, either the convection term $\bar{\boldsymbol{v}}\cdot\bar{\nabla}\bar{u}$ or the diffusion term $\bar{\nabla}^2\bar{u}$ must deviate significantly from unity.

Consider the following 1D example to illustrate the point: $\boldsymbol{v} = v\boldsymbol{i}$, $v > 0$ constant, a domain $[0, L]$, with boundary conditions $u(0) = 0$ and $u(L) = U_L$,

$$vu' = \alpha u'', \quad u(0) = 0,\ u(L) = U_L\,.$$

The scaled problem reads

$$\frac{d\bar{u}}{d\bar{x}} = \mathrm{Pe}^{-1}\frac{d^2\bar{u}}{d\bar{x}^2}, \quad \bar{x}\in(0,1), \quad \bar{u}(0) = 0,\ \bar{u}(1) = 1,$$

if we choose $U = U_L$. The solution of the scaled problem is

$$\bar{u}(\bar{x}) = \frac{1 - e^{\bar{x}\mathrm{Pe}}}{1 - e^{\mathrm{Pe}}}\,.$$

Figure 13 indicates how $\bar{u}$ depends on Pe: small Pe values gives approximately a straight line while large Pe values leads to a *boundary layer* close to $x = 1$, where the solution changes very rapidly.

We realize that for large Pe,

$$\max_{\bar{x}}\frac{d\bar{u}}{d\bar{x}} \approx \mathrm{Pe}, \quad \max_{\bar{x}}\frac{d^2\bar{u}}{d\bar{x}^2} \approx \mathrm{Pe}^2,$$

which are consistent results with the PDE since the double derivative term is multiplied by Pe^{-1}. For small Pe,

$$\max_{\bar{x}}\frac{d\bar{u}}{d\bar{x}} \approx 1, \quad \max_{\bar{x}}\frac{d^2\bar{u}}{d\bar{x}^2} \approx 0,$$

which is also consistent with the PDE since an almost vanishing second-order derivative is multiplied by a very large coefficient Pe^{-1}. However, we have a problem with very large derivatives of $\bar{u}$ when Pe is large.

To arrive at a proper scaling for large Peclet numbers, we need to remove the Pe coefficient from the differential equation. There are only two scales at our disposals: u_c and x_c for u and x, respectively. The natural value for u_c is the boundary value U_L at $x = L$. The scaling of $Vu_x = \alpha u_{xx}$ then results in

$$\frac{d\bar{u}}{d\bar{x}} = \frac{\alpha}{Vx_c}\frac{d^2\bar{u}}{d\bar{x}^2}, \quad \bar{x}\in(0,\bar{L}), \quad \bar{u}(0) = 0,\ \bar{u}(\bar{L}) = 1,$$

where $\bar{L} = L/x_c$. Choosing the coefficient $\alpha/(Vx_c)$ to be unity results in the scale $x_c = \alpha/V$, and $\bar{L}$ becomes Pe. The final, scaled boundary-value problem is now

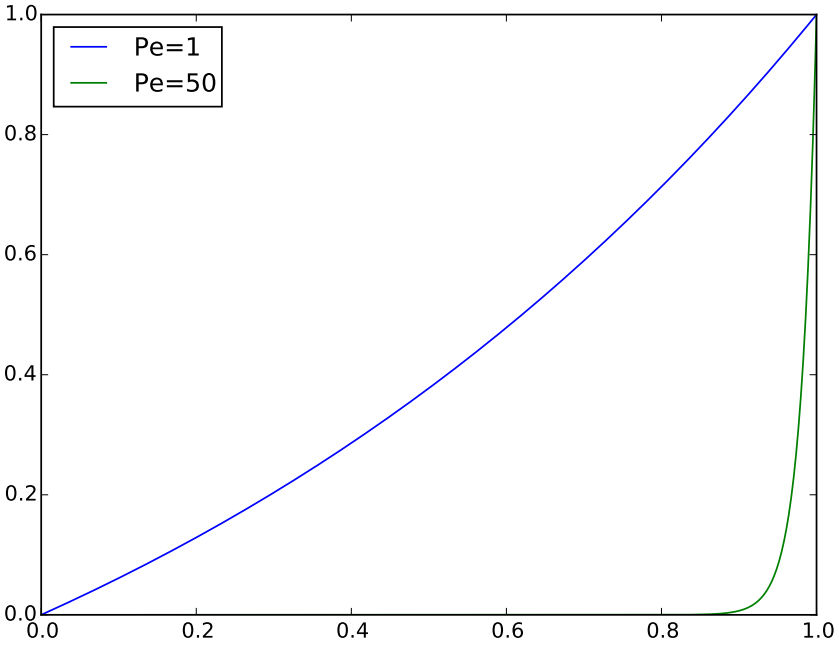

Figure 13: Solution of scaled problem where the length scale depends on the Peclet number.

$$\frac{d\bar{u}}{d\bar{x}} = \frac{d^2\bar{u}}{d\bar{x}^2}, \quad \bar{x} \in (0, \mathrm{Pe}), \quad \bar{u}(0) = 0,\ \bar{u}(\mathrm{Pe}) = 1,$$

with solution

$$\bar{u}(\bar{x}) = \frac{1 - e^{\bar{x}}}{1 - e^{\mathrm{Pe}}}\,.$$

Figure 14 displays $\bar{u}$ for some Peclet numbers, and we see that the shape of the graphs are the same with this scaling. For large Peclet numbers we realize that $\bar{u}$ and its derivatives are around unity ($1 - e^{\mathrm{Pe}} \approx -e^{\mathrm{Pe}}$), but for small Peclet numbers $d\bar{u}/d\bar{x} \sim \mathrm{Pe}^{-1}$.

The conclusion is that for small Peclet numbers, $x_c = L$ is an appropriate length scale and leads to scaled derivatives in $[0, 1]$. The scaled equation $\mathrm{Pe}\bar{u}' = \bar{u}''$ indicates that $\bar{u}'' \approx 0$, and the solution is close to a straight line. For large Pe values, $x_c = \alpha/V$ is an appropriate length scale, and the scaled equation expresses that the terms $\bar{u}'$ and $\bar{u}''$ are equal and of size around unity.

6.3 Convection-diffusion with a force term

Let us add a force term $f(\boldsymbol{x}, t)$ to the convection-diffusion equation :

$$\frac{\partial u}{\partial t} + \boldsymbol{v} \cdot \nabla u = \alpha \nabla^2 u + f\,. \tag{163}$$

The scaled version reads

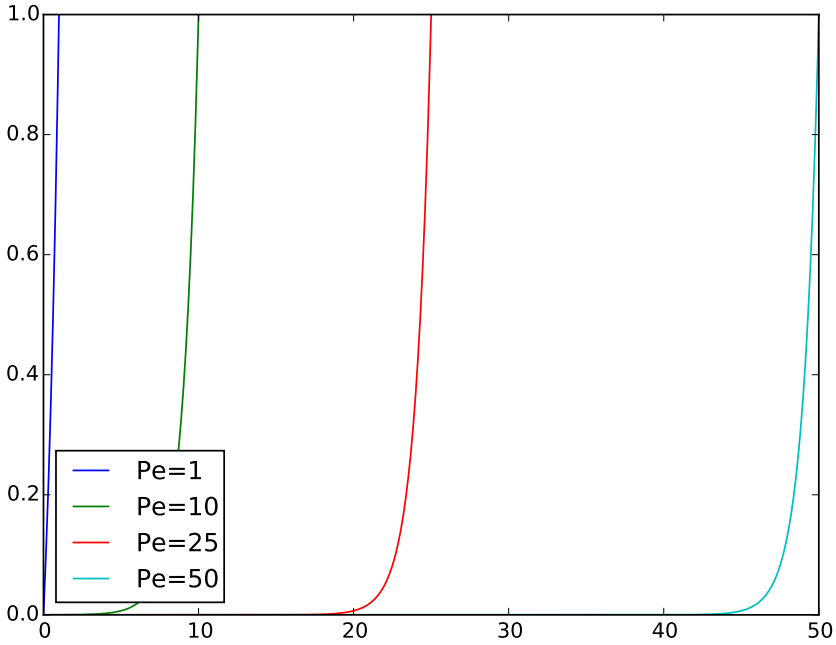

Figure 14: Solution of scaled problem where the length scale depends on the Peclet number.

$$\frac{\partial u}{\partial t} + \frac{t_c V}{L}\bar{\boldsymbol{v}}\cdot\bar{n}abla\bar{u} = \frac{t_c\alpha}{L^2}\bar{\nabla}^2\bar{u} + \frac{t_c f_c}{u_c}\bar{f}\,.$$

We can base t_c on convective transport: $t_c = L/V$. Now, u_c could be chosen to make the coefficient in the source term unity: $u_c = t_c f_c = Lf_c/V$. This leaves us with

$$\frac{\partial u}{\partial t} + \bar{\boldsymbol{v}}\cdot\bar{\nabla}\bar{u} = \mathrm{Pe}^{-1}\bar{\nabla}^2\bar{u} + \bar{f}\,.$$

In the diffusion limit, we base t_c on the diffusion time scale: $t_c = L^2/\alpha$, and the coefficient of the source term set to unity determines u_c according to

$$\frac{L^2 f_c}{\alpha u_c} = 1 \quad \Rightarrow \quad u_c = \frac{L^2 f_c}{\alpha}\,.$$

The corresponding PDE reads

$$\frac{\partial u}{\partial t} + \mathrm{Pe}\,\bar{\boldsymbol{v}}\cdot\bar{\nabla}\bar{u} = \bar{\nabla}^2\bar{u} + \bar{f},$$

so for small Peclet numbers, which we have, the convective term can be neglected and we get a pure diffusion equation with a source term.

So, what if the problem is stationary? Then there is no time scale and we get

$$\frac{Vu_c}{L}\bar{\boldsymbol{v}}\cdot\bar{\nabla}\bar{u} = \frac{u_c\alpha}{L^2}\bar{\nabla}^2\bar{u} + f_c\bar{f},$$

or

$$\bar{\boldsymbol{v}} \cdot \bar{\nabla} \bar{u} = \mathrm{Pe}^{-1} \bar{\nabla}^2 \bar{u} + \frac{f_c L}{V u_c} \bar{f},$$

Again, choosing u_c such that the source term coefficient is unity leads to $u_c = f_c L / V$.

hpl 8: Suppose V is small. This suggests removing the convection *and* the source term.

7 The equations of linear elasticity

Sketch:

$$\varrho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} = \nabla((\lambda + \mu) \nabla \cdot \boldsymbol{u}) + \nabla \cdot (\mu \nabla \boldsymbol{u}) + \varrho \boldsymbol{f} \,. \tag{164}$$

$$\varrho \frac{\partial^2 \bar{\boldsymbol{u}}}{\partial \bar{t}^2} = \bar{\nabla}((\frac{t_c^2 \lambda_c}{\varrho L^2} \bar{\lambda} + \frac{t_c^2 \mu_c}{\varrho L^2} \mu) \bar{\nabla} \cdot \bar{\boldsymbol{u}}) + \frac{t_c^2 \mu_c}{\varrho L^2} \bar{\nabla} \cdot (\bar{\mu} \bar{\nabla} \bar{\boldsymbol{u}}) + \frac{t_c^2 f_c}{u_c} \bar{\boldsymbol{f}} \,. \tag{165}$$

$$t_c = L \sqrt{\frac{\varrho}{\mu_c}}$$

$$u_c = \frac{\varrho L^2 f_c}{\mu_c}$$

$$\varrho \frac{\partial^2 \bar{\boldsymbol{u}}}{\partial \bar{t}^2} = \bar{\nabla}((\alpha \bar{\lambda} + \bar{\mu}) \bar{\nabla} \cdot \bar{\boldsymbol{u}}) + \bar{\nabla} \cdot (\bar{\mu} \bar{\nabla} \bar{\boldsymbol{u}}) + \boldsymbol{f} \,. \tag{166}$$

$$\alpha = \frac{\lambda_c}{\mu_c} \,.$$

hpl 9: The homogeneous case without body force is common. Scale displacement from boundary condition. Maybe scale traction first and determine characteristic displacement from it?

hpl 10: The homogeneous case should be written up, then $\mu_c = \mu$ and $\bar{\mu} = 1$, etc.

8 The Navier-Stokes equations

8.1 The momentum equation without body forces

The Navier-Stokes equations for incompressible viscous fluid flow take the form

$$\varrho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \mu \nabla^2 \boldsymbol{u}, \tag{167}$$

$$\nabla \cdot \boldsymbol{u} = 0 \,. \tag{168}$$

The primary unknowns are the $\boldsymbol{u}$ is the velocity of the fluid and p is the pressure, while ϱ is the fluid density, and μ is the dynamic viscosity.

We introduce as usual dimensionless independent and dependent variables:

$$\bar{x} = \frac{x}{L}, \quad \bar{y} = \frac{y}{L}, \quad \bar{z} = \frac{z}{L}, \quad \bar{t} = \frac{t}{t_c}, \quad \bar{\boldsymbol{u}} = \frac{\boldsymbol{u}}{u_c}, \quad \bar{p} = \frac{p}{p_c},$$

where L is some characteristic distance, t_c is some characteristic time, u_c is a characteristic velocity, and p_c is characteristic pressure. Inserted in the equations,

$$\varrho\left(\frac{u_c}{t_c}\frac{\partial\bar{\boldsymbol{u}}}{\partial\bar{t}} + \frac{u_c^2}{L}\bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{\boldsymbol{u}}\right) = -\frac{p_c}{L}\bar{\nabla}\bar{p} + \frac{u_c}{L^2}\mu\bar{\nabla}^2\bar{\boldsymbol{u}}, \tag{169}$$

$$\frac{u_c}{L}\bar{\nabla}\cdot\bar{\boldsymbol{u}} = 0\,. \tag{170}$$

For the velocity it is common to just introduce some U for u_c. This U is normally implied by the problem description. **hpl 11**: Show examples of confined and unconfined flow and what U is. For example, U is often chosen as a characteristic inlet velocity in the flow problem. Having a characteristic distance L and velocity U, an obvious time measure is L/U so we set $t_c = L/U$. Dividing by the coefficient in front of the time derivative term, creates a pressure term

$$\frac{p_c}{\varrho U^2}\bar{\nabla}\bar{p}\,.$$

The coefficient suggest a choice $p_c = \varrho U^2$ if the pressure gradient term is to have the same size as the acceleration terms.

8.2 The most common dimensionless form of the Navier-Stokes equations

The discussions so far results in the following dimensionless form of (167) and (168):

$$\frac{\partial\bar{\boldsymbol{u}}}{\partial\bar{t}} + \bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{\boldsymbol{u}} = -\bar{\nabla}p + \mathrm{Re}^{-1}\nabla^2\boldsymbol{u}, \tag{171}$$

$$\bar{\nabla}\cdot\bar{\boldsymbol{u}} = 0, \tag{172}$$

where Re is the famous *Reynolds number*,

$$\mathrm{Re} = \frac{\varrho UL}{\mu} = \frac{UL}{\nu}\,.$$

The latter expression makes use of the kinematic viscosity $\nu = \mu/\varrho$. For viscous fluid flows without body forces there is hence only one dimensionless number, Re.

The Reynolds number can be interpreted as the ratio of convection and viscosity:

$$\mathrm{Re} = \frac{\text{convection}}{\text{viscosity}} = \frac{\varrho U^2/L}{\mu U/L^2} = \frac{UL}{\nu}\,.$$

(We have here used that $\nabla\boldsymbol{u}$ goes like U/L and $\nabla^2\boldsymbol{u}$ goes like U/L^2.)

8.3 Scaling of time for low Reynolds numbers

As we discussed in Section 6.3 for the convection-diffusion equation, there is not just one scaling that fits all problems. Above, we used $t_c = L/U$, which is appropriate if convection is a dominating physical effect. In case the convection term $\varrho\boldsymbol{u}\cdot\nabla\boldsymbol{u}$ is much smaller than the viscosity term $\mu\nabla^2\boldsymbol{u}$, i.e., the Reynolds number is small, the viscosity term is dominating. However, if the scaling is suitable, the other terms are of order unity, and $\mathrm{Re}^{-1}\nabla^2\bar{\boldsymbol{u}}$ must then also be of unit size, implying that $\nabla^2\bar{\boldsymbol{u}}$ is very small, but then the scaling is not suitable. In the low-Reynolds number regime, the diffusion effect of $\nabla^2\bar{\boldsymbol{u}}$ is dominating, and we should use a time scale based

on diffusion rather than convection. Such a time scale is $t_c = L^2/(\mu/\varrho) = L^2/\nu$. With this time scale, the Navier-Stokes equation looks like

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial \bar{t}} + \mathrm{Re}\,\bar{\boldsymbol{u}} \cdot \bar{\nabla}\bar{\boldsymbol{u}} = -\bar{\nabla}p + \nabla^2 \boldsymbol{u}, \tag{173}$$

$$\bar{\nabla} \cdot \bar{\boldsymbol{u}} = 0\,. \tag{174}$$

As stated in the box in Section 6.3, (173) is the appropriate PDE for very low Reynolds number flow and suggest neglecting the convection term. If the flow is also steady, the time derivative term can be neglected, and we end up with the so-called *Stokes problem* for steady, slow, viscous flow:

$$-\bar{\nabla}p + \nabla^2 \boldsymbol{u} = 0, \tag{175}$$

$$\bar{\nabla} \cdot \bar{\boldsymbol{u}} = 0\,. \tag{176}$$

This flow regime is also known as *Stokes' flow* or *creeping flow*.

8.4 Shear stress as pressure scale

Instead of using the kinetic energy ϱU^2 as pressure scale, one can use the shear stress $\mu U/L$ (U/L reflects the spatial derivative of the velocity, as in the shear stress expression $\mu \partial u/\partial y$). Using U as velocity scale, L/U as time scale, and $\mu U/L$ as pressure scale, results in

$$\mathrm{Re}\left(\frac{\partial \bar{\boldsymbol{u}}}{\partial \bar{t}} + \bar{\boldsymbol{u}} \cdot \bar{\nabla}\bar{\boldsymbol{u}}\right) = -\bar{\nabla}\bar{p} + \nabla^2 \bar{\boldsymbol{u}}\,. \tag{177}$$

8.5 Including the gravity force

We now add a gravity force to the momentum equation (167):

$$\varrho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) = -\nabla p + \mu \nabla^2 \boldsymbol{u} - \varrho g \boldsymbol{k}, \tag{178}$$

where g is the acceleration of gravity, and $\boldsymbol{k}$ is a unit vector in the opposite direction of gravity. The new term takes the following form after non-dimensionalization:

$$\frac{t_c}{\varrho u_c}\varrho g \boldsymbol{k} = \frac{Lg}{U^2}\boldsymbol{k} = \mathrm{Fr}^{-2}\boldsymbol{k},$$

where Fr is the Froude number,

$$\mathrm{Fr} = \frac{U}{\sqrt{Lg}}\,.$$

8.6 Oscillating boundary conditions

Many flows has an oscillating nature, often arising from some oscillating boundary conditions. Suppose such a condition at some boundary $x = \text{const}$ takes the specific form

$$\boldsymbol{u} = U \sin(\omega t)\boldsymbol{i}\,.$$

The dimensionless form becomes

$$U\bar{\boldsymbol{u}} = U\sin(\omega\frac{L}{U}\bar{t})\boldsymbol{i},$$

if $t_c = L/U$ is the appropriate time scale. This condition can be written

$$\bar{\boldsymbol{u}} = \sin(\mathrm{St}\,\bar{t}), \tag{179}$$

where St is the *Strouhal number*,

$$\mathrm{St} = \frac{\omega L}{U}\,. \tag{180}$$

The two important dimensionless parameters in oscillating flows are then the Reynolds and Strouhal numbers.

Even if the boundary conditions are of steady type, as for flow around a sphere or cylinder, the flow may at certain Reynolds numbers get unsteady and oscillating. For $10^2 < \mathrm{Re} < 10^7$, steady inflow towards a cylinder will cause vortex shedding: an array of vortices are periodically shedded from the cylinder, producing an oscillating flow pattern and force on the cylinder. The Strouhal number is used to characterize the frequency of oscillations. The phenomenon, known as *von Karman vortex street*, is particularly important if the frequency of the force on the cylinder hits the free vibration frequency of the cylinder such that resonance occurs. The result can be large displacements of the cylinder and structural failure. A famous case in engineering is the failure of the Tacoma Narrows suspension bridge[20] in 1940, when wind-induced vortex shedding caused resonance with the free torsional vibrations of the bridge.

8.7 The Euler number

The dimensionless pressure in (171) made use of the pressure scale $p_c = \varrho U^2$. This is an appropriate scale if the pressure level is not of importance, which is very often the case since only the pressure *gradient* enters the flow equation and drives the flow. However, there are circumstances where the pressure level is of importance. For example, in some flows the pressure may become so low that the vapor pressure of the liquid is reached and that vapor cavities form (a phenomenon known as *cavitation*). A more appropriate pressure scale is then $p_c = p_\infty - p_v$, where p_∞ is a characteristic pressure level far from vapor cavities and p_v is the vapor pressure. The coefficient in front of the dimensionless pressure gradient is then

$$\frac{p_\infty - p_v}{\varrho U^2}\,.$$

Inspired by Bernoulli's equation $p + \frac{1}{2}\varrho U^2 = \text{const}$ in fluid mechanics, a factor $\frac{1}{2}$ is often inserted in the denominator, and the corresponding dimensionless number,

$$\mathrm{Eu} = \frac{p_\infty - p_v}{\frac{1}{2}\varrho U^2}, \tag{181}$$

is called the *Euler number*. The pressure gradient term now reads $\frac{1}{2}\mathrm{Eu}\,\bar{\nabla}\bar{p}$. The Euler number expresses the ratio of pressure differences and the kinetic energy of the flow.

[20] `https://en.wikipedia.org/wiki/Tacoma_Narrows_Bridge_(1940)`

8.8 Free surface conditions

At a free surface, $z = \eta(x, y, t)$, the boundary conditions are

$$w = \frac{\partial \eta}{\partial t} + \boldsymbol{u} \cdot \nabla \eta, \tag{182}$$

$$p - p_0 = -\sigma \left(R_x^{-1} + R_y^{-1} \right) \approx -\sigma \left(\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} \right), \tag{183}$$

where w is the velocity component in the z direction, p_0 is the atmospheric air pressure at the surface, σ represents the surface tension, while R_x and R_y are radii of curvature of the surface $z = \eta$, which for small surface deformations can be approximated by second-order derivatives of η as indicated.

The dimensionless form of these conditions starts with

$$u_c \bar{w} = \frac{L}{t_c} \frac{\partial \bar{\eta}}{\partial \bar{t}} + u_c \bar{\boldsymbol{u}} \cdot \bar{\nabla} \bar{\eta},$$

$$p_c \bar{p} \approx -\frac{1}{L} \sigma \left(\frac{\partial^2 \bar{\eta}}{\partial \bar{x}^2} + \frac{\partial^2 \bar{\eta}}{\partial \bar{y}^2} \right).$$

The characteristic length L is usually taken as the depth of the fluid when the surface is flat. We have used $\bar{p} = (p - p_0)/p_c$ for making the pressure dimensionless. Using $u_c = U$, $t_c = L/U$, and $p_c = \varrho U^2$, results in

$$\bar{w} = \frac{\partial \bar{\eta}}{\partial \bar{t}} + \bar{\boldsymbol{u}} \cdot \bar{\nabla} \bar{\eta}, \tag{184}$$

$$\bar{p} \approx -\mathrm{We}^{-1} \left(\frac{\partial^2 \bar{\eta}}{\partial \bar{x}^2} + \frac{\partial^2 \bar{\eta}}{\partial \bar{y}^2} \right), \tag{185}$$

where We is the *Weber number*,

$$\mathrm{We} = \frac{\varrho U^2 L}{\sigma}. \tag{186}$$

The weber number measures the importance of surface tension effects and is the ratio of the pressure scale ϱU^2 and the surface tension force per area, typically σ / R_x in a 2D problem, which has size σ / L.

9 Thermal convection

Temperature differences in fluid flow cause density differences, and since cold fluid is heavier than hot fluid, the gravity force will induce flow due to density differences. This effect is called free thermal convection. Forced convection refers to the case where there is no feedback from the temperature field to the motion, i.e., temperature differences do not create motion, and the temperature distribution is only dependent on a given velocity field.

9.1 Forced convection

The model governing forced convection consists of the Navier-Stokes equation and the energy equation for the temperature:

$$\varrho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \mu\nabla^2\boldsymbol{u} - \varrho g\boldsymbol{k}, \tag{187}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{188}$$

$$\varrho c\left(\frac{\partial T}{\partial t} + \boldsymbol{u}\cdot\nabla T\right) = \kappa\nabla^2 T\,. \tag{189}$$

$$\tag{190}$$

The symbol T is the temperature, c is a heat capacity, and κ is the heat conduction coefficient for the fluid.

Despite the fact that ϱ depends on T, we treat ϱ as a constant. The major effect of this dependence is through the buoyancy effect caused by the gravity term $-g\boldsymbol{k}$. We drop this term, and assume the momentum and continuity equations to be independent of the temperature. The flow is driven by boundary conditions, from which we can find a characteristic velocity U.

We introduce dimensionless parameters according to

$$\bar{x} = \frac{x}{L},\ t_c = \frac{L}{U},\ \bar{\boldsymbol{u}} = \frac{\boldsymbol{u}}{U},\ \bar{p} = \frac{p}{\varrho U^2},\ \bar{T} = \frac{T - T_0}{T_c}\,.$$

Other coordinates are also scaled by L. The characteristic temperature T_c is chosen as some range ΔT, which depends on the problem and is often given by the thermal initial and/or boundary conditions. The reference temperature T_0 is also implied by prescribed conditions. Inserted in the equations, we get

$$\begin{aligned}
\varrho\frac{U^2}{L}\frac{\partial\bar{\boldsymbol{u}}}{\partial\bar{t}} + \varrho\frac{U^2}{L}\bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{\boldsymbol{u}} &= -1L\bar{\nabla}\bar{p} + \frac{\mu U}{L^2}\bar{\nabla}^2\bar{\boldsymbol{u}},\\
\frac{U}{L}\bar{\nabla}\cdot\bar{\boldsymbol{u}} &= 0,\\
\varrho_0 c\left(\frac{T_c U}{L}\frac{\partial\bar{T}}{\partial\bar{t}} + \frac{UT_c}{L}\bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{T}\right) &= \frac{\kappa T_c}{L^2}\bar{\nabla}^2\bar{T}\,.
\end{aligned}$$

Making each term in each equation dimensionless reduces the system to

$$\frac{\partial\bar{\boldsymbol{u}}}{\partial\bar{t}} + \bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{\boldsymbol{u}} = -\bar{\nabla}\bar{p} + \mathrm{Re}^{-1}\bar{\nabla}^2\bar{\boldsymbol{u}}, \tag{191}$$

$$\bar{\nabla}\cdot\bar{\boldsymbol{u}} = 0, \tag{192}$$

$$\frac{\partial\bar{T}}{\partial\bar{t}} + \bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{T} = \mathrm{Pe}^{-1}\bar{\nabla}^2\bar{T}\,. \tag{193}$$

The two dimensionless numbers in this system are given by

$$\mathrm{Pe} = \frac{\varrho_0 cUL}{\kappa},\quad \mathrm{Re} = \frac{UL}{\nu}\,.$$

The Peclet number is here defined as the ratio of the convection term for heat $\varrho_0 cU\Delta T/L$ and the heat conduction term $\kappa U/L^2$. The fraction $\kappa/(\varrho_0 c)$ is known as the thermal diffusivity, and if this quantity is given a symbol α, we realize the relation to the Peclet number defined in Section 6.3.

9.2 Free convection

hpl 12: Do the general model and simplified models, e.g., free convection near a wall.

Governing equations. The mathematical model for free thermal convection consists of the Navier-Stokes equations coupled to an energy equation governing the temperature:

$$\varrho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \mu\nabla^2\boldsymbol{u} - \varrho g\boldsymbol{k}, \tag{194}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{195}$$

$$\varrho c\left(\frac{\partial T}{\partial t} + \boldsymbol{u}\cdot\nabla T\right) = \kappa\nabla^2 T + 2\mu\varepsilon_{ij}\varepsilon_{ij}\,. \tag{196}$$

$$\tag{197}$$

The symbol T is the temperature, c is a heat capacity, κ is the heat conduction coefficient for the fluid. In free convection, the gravity term $-g\boldsymbol{k}$ is essential since the flow is driven by temperature differences and the fact that hot fluid rises while cold fluid falls.

Heating by viscous effects. We have also included heating of the fluid due to viscous forces through the term $2\mu\varepsilon_{ij}\varepsilon_{ij}$, where ε_{ij} is the strain-rate tensor in the flow, defined by

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = \frac{1}{2}(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T),$$

where u_i is the velocity in direction of x_i ($i = 1, 2, 3$ measures the space directions). The term $2\mu\varepsilon_{ij}\varepsilon_{ij}$ is written with Einstein's summation convention in mind such that there is an implicit sum over i and j. This term is also relevant for forced convection, but was left out in Section 9.2 for mathematical simplicity. However, heating by the work of is often a very small effect and can be neglected, although it plays a major role in forging and extrusion of metals. By making the temperature equation dimensionless, we will get a precise measure when the term can be neglected.

Relation between density and temperature. The equations (194) and (195) has already been made dimensionless in the previous section. The major difference is now that ϱ is no longer a constant, but a function of T. The relationship between ϱ and T is often taken as linear,

$$\varrho = \varrho_0 - \varrho_0\beta(T - T_0),$$

where

$$\beta = -\frac{1}{\varrho}\left(\frac{\partial\varrho}{\partial t}\right)_p,$$

is known as the thermal expansion coefficient of the fluid, and ϱ_0 a reference density when the temperature is at T_0.

Comment on the form of the equation of continuity. It might look strange that the equation of continuity (from the mass conservation principle) is $\nabla \cdot \boldsymbol{u} = 0$ when there are density variations in the flow. The rationale for using this version of the continuity equation is the assumption that the density of each fluid particle remains constant. Consequently, from the general equation of continuity,

$$\frac{\partial \varrho}{\partial t} + \nabla \cdot (\varrho \boldsymbol{u}) = \frac{D\varrho}{dt} + \varrho \nabla \cdot \boldsymbol{u} = 0,$$

it follows that if ϱ is constant for a particle, the material derivative $D\varrho/dt = \varrho_t + \boldsymbol{v} \cdot \nabla \varrho = 0$, and the equation reduces to $\nabla \cdot \boldsymbol{u} = 0$.

The Boussinesq approximation. A very common approximation, called the *Boussinesq approximation*, is to neglect the density variations in all terms except the gravity term. This is a good approximation unless the change is ϱ is large. With the linear $\varrho(T)$ formula and the Boussinesq approximation, (194)-(196) take the form

$$\varrho_0 \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \mu \nabla^2 \boldsymbol{u} - (\varrho_0 - \varrho_0 \beta (T - T_0)) g \boldsymbol{k}, \tag{198}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{199}$$

$$\varrho_0 c \left(\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T \right) = \kappa \nabla^2 T + 2\mu \varepsilon_{ij} \varepsilon_{ij} \, . \tag{200}$$

$$\tag{201}$$

A good justification of the Boussinesq approximation is provided by Tritton [8, Ch. 13].

Scaling. Dimensionless variables are introduced as

$$\bar{x} = \frac{x}{L}, \quad t_c = \frac{L}{U}, \; \bar{\boldsymbol{u}} = \frac{\boldsymbol{u}}{U}, \; \bar{p} = \frac{p}{\varrho U^2}, \; \bar{T} = \frac{T - T_0}{\Delta T} \, .$$

The dimensionless y and z coordinates also make use of L as scale. As in forced convection, we assume the characteristic temperature level T_0 and the scale ΔT is given by thermal boundary and/or initial conditions. Contrary to Sections 8 and 9.2, U is now not given by the problem description, but implied by ΔT.

Replacing quantities with dimensions by their dimensionless counterparts results in

$$\varrho_0 \frac{U^2}{L} \frac{\partial \bar{\boldsymbol{u}}}{\partial \bar{t}} + \varrho_0 \frac{U^2}{L} \bar{\boldsymbol{u}} \cdot \bar{\nabla} \bar{\boldsymbol{u}} = -\frac{p_c}{L} \bar{\nabla} \bar{p} + \frac{\mu U}{L^2} \bar{\nabla}^2 \bar{\boldsymbol{u}} - \varrho_0 g \boldsymbol{k} + \varrho_0 \beta T_c \bar{T} g \boldsymbol{k},$$

$$\frac{U}{L} \bar{\nabla} \cdot \bar{\boldsymbol{u}} = 0,$$

$$\varrho_0 c \left(\frac{T_c U}{L} \frac{\partial \bar{T}}{\partial \bar{t}} + \frac{U T_c}{L} \bar{\boldsymbol{u}} \cdot \bar{\nabla} \bar{T} \right) = \frac{\kappa T_c}{L^2} \bar{\nabla}^2 \bar{T} + 2 \frac{\mu U}{L} \bar{\varepsilon}_{ij} \bar{\varepsilon}_{ij} \, .$$

These equations reduce to

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial \bar{t}} + \bar{\boldsymbol{u}} \cdot \bar{\nabla} \bar{\boldsymbol{u}} = -\bar{\nabla} \bar{p} + \mathrm{Re}^{-1} \bar{\nabla}^2 \bar{\boldsymbol{u}} - \mathrm{Fr}^{-2} \boldsymbol{k} + \gamma \bar{T} \boldsymbol{k}, \tag{202}$$

$$\bar{\nabla} \cdot \bar{\boldsymbol{u}} = 0, \tag{203}$$

$$\frac{\partial \bar{T}}{\partial \bar{t}} + \bar{\boldsymbol{u}} \cdot \bar{\nabla} \bar{T} = \mathrm{Pe}^{-1} \bar{\nabla}^2 \bar{T} + 2\delta \bar{\varepsilon}_{ij} \bar{\varepsilon}_{ij} \,. \tag{204}$$

The dimensionless numbers are given by

$$\gamma = \frac{g\beta L \Delta T}{U^2}, \quad \mathrm{Pe}^{-1} = \frac{\kappa}{\varrho_0 c U L}, \quad \delta = \frac{\mu U}{L \varrho_0 c \Delta T} \,.$$

The Peclet number is here defined as the ratio of the convection term for heat $\varrho_0 c U \Delta T / L$ and the heat conduction term $\kappa U / L^2$. The γ number measures the ratio of thermal buoyancy and the convection term:

$$\gamma = \frac{\varrho_0 g \beta \Delta T}{\varrho_0 U^2 / L} = \frac{g \beta L \Delta T}{U^2} \,.$$

The Pe parameter is the fraction of the convection term and the thermal diffusion term:

$$\frac{\varrho_0 c U \Delta T L^{-1}}{\kappa L^{-2} \Delta T} = \frac{\varrho c U L}{\kappa} = \mathrm{Pe} \,.$$

The δ parameter is the ratio of the viscous dissipation term and the convection term:

$$\delta = \frac{\mu U^2 / L^2}{\varrho_0 c U \Delta T / L} == \frac{\mu U}{L \varrho_0 c \Delta T} \,.$$

9.3 The Grashof, Prandtl, and Eckert numbers

The problem with the above dimensionless numbers is that they involve U, but U is implied by ΔT. Assuming that the convection term is much bigger than the viscous diffusion term, the momentum equation features a balance between the buoyancy term and the convection term:

$$|\varrho_0 \boldsymbol{u} \cdot \nabla \boldsymbol{u}| \sim \varrho_0 g \beta \Delta T \,.$$

Translating this similarity to scales,

$$\varrho_0 U^2 / L \sim \varrho_0 g \beta \Delta T,$$

gives an U of in terms of ΔT :

$$U = \sqrt{\beta L \Delta T} \,.$$

The Reynolds number with this U now becomes

$$\mathrm{Re}_T = \frac{UL}{\nu} = \frac{\sqrt{g \beta L^3 \Delta T}}{\nu^2} = \mathrm{Gr}^{1/2},$$

where Gr is the Grashof number in free thermal convection:

$$\mathrm{Ga} = \mathrm{Re}_T^2 = \frac{g \beta L^3 \Delta T}{\nu^2} \,.$$

The Grashof number replaces the Reynolds number in the scaled equations of free thermal convection. We shall soon look at its interpretations, which are not as straightforward as for the Reynolds and Peclet numbers.

The above choice of U in terms of ΔT results in γ equal to unity:

$$\gamma = \frac{g\beta L\Delta T}{U^2} = \frac{g\beta L\Delta T}{g\beta L\Delta T} = 1\,.$$

The Peclet number can also be rewritten as

$$\mathrm{Pe} = \frac{\varrho c U L}{\kappa} = \frac{\mu c}{\kappa}\frac{\varrho U L}{\mu} = \mathrm{PrRe}^{-1} = \mathrm{PrRe}_T,$$

where Pr is the Prandtl number, defined as

$$\mathrm{Pr} = \frac{\mu c}{\kappa}\,.$$

The Prandtl number is the ratio of the momentum diffusivity (kinematic viscosity) and the thermal diffusivity. Actually, analysis show that Pr reflects the ratio of the thickness of the thermal and velocity boundary layers: when $\mathrm{Pr} = 1$, these layers coincide, while $\mathrm{Pr} \ll 1$ implies that the thermal layer is much thicker than the velocity boundary layer, and vice versa for $\mathrm{Pe} \gg 1$.

The δ parameter is in free convection replaced by a combination of the Eckert number (Ec) and the Reynolds number. We have that

$$\mathrm{Ec} = \frac{U^2}{c\Delta T} = \delta \mathrm{Re}_T,$$

and consequently

$$\delta = \mathrm{EcRe}_T^{-1}\,.$$

Writing

$$\mathrm{Ec} = \frac{\varrho_0 U^2}{\varrho_0 c\Delta T},$$

shows that the Eckert number can be interpreted as the ratio of the kinetic energy of the flow and the thermal energy.

We use Ga instead of Re in the momentum equations and also instead of Pe in the energy equation (recall that $\mathrm{Pe} = \mathrm{PrRe} = \mathrm{PrRe}_T = \mathrm{PrGr}^{-1/2}$). The resulting scaled system becomes

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial \bar{t}} + \bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{\boldsymbol{u}} = -\bar{\nabla}\bar{p} + \mathrm{Gr}^{-1/2}\bar{\nabla}^2\bar{\boldsymbol{u}} - \mathrm{Fr}^{-2}\boldsymbol{k} + \bar{T}\boldsymbol{k}, \tag{205}$$

$$\bar{\nabla}\cdot\bar{\boldsymbol{u}} = 0, \tag{206}$$

$$\mathrm{Gr}^{1/2}\left(\frac{\partial \bar{T}}{\partial \bar{t}} + \bar{\boldsymbol{u}}\cdot\bar{\nabla}\bar{T}\right) = \mathrm{Pr}^{-1}\bar{\nabla}^2\bar{T} + 2\mathrm{EcRe}_T^{-1}\bar{\varepsilon}_{ij}\bar{\varepsilon}_{ij}\,. \tag{207}$$

We realize that in free convection, the Grashof number plays the same role as the Reynolds number in the momentum equation. In particular, it turns out that Gr governs the transition between laminar and turbulent flow. For example, the transition to turbulence occurs in the range $10^8 < \mathrm{Gr} < 10^9$ for free convection from vertical flat plates. Gr is normally interpreted as a dimensionless number expressing the ratio of buoyancy forces and viscous forces.

Recall that the scaling leading to the Grashof number is based on an estimate of U from a balance of the convective and the buoyancy terms. When the viscous term dominates over convection, we need a different estimate of U, since in this case, the viscous force balances the buoyancy force:

$$\mu\nabla^2\boldsymbol{u} \sim \varrho_0 g\beta\Delta T \quad \Rightarrow \quad \mu U/L^2 \sim \varrho_0 g\beta\Delta T,$$

This similarity suggests the scale

$$U = \frac{g\beta L^2\Delta T}{\nu}\,.$$

Now,

$$\frac{|\varrho_0\boldsymbol{u}\cdot\nabla\boldsymbol{u}|}{|\mu\nabla^2\boldsymbol{u}|} = \frac{UL}{\nu} = \frac{g\beta L^3\Delta T}{\nu} = \mathrm{Gr}\,.$$

The result means that $\mathrm{Gr}^{1/2}$ measures the ratio of convection and viscous forces when convection dominates, but Gr measures this ratio when viscous forces dominate.

hpl 13: Normally, convection dominates in free convection!

hpl 14: Is it not so that Ra is interesting primarily in horizontal layers? Could appear naturally when scaling simplified equations?

The product of Gr and Pr is the Rayleigh number,

$$\mathrm{Ra} = \frac{g\beta L^3\Delta T\varrho_0 c}{\nu\kappa},$$

since

$$\mathrm{GrPr} = \mathrm{Re}_T^2\mathrm{Pr} = \frac{g\beta L^3\Delta T}{\nu^2}\frac{\mu c}{\kappa} = \frac{g\beta L^3\Delta T\varrho_0 c}{\nu\kappa} = \mathrm{Ra}\,.$$

hpl 15: Need an example using Ra.

9.4 Heat transfer at boundaries

A common boundary condition, modeling heat transfer to/from the surroundings is

$$-\kappa\frac{\partial T}{\partial n} = h_T(T - T_s), \tag{208}$$

where $\partial/\partial n$ means derivative in the normal direction $(\boldsymbol{n}\cdot\nabla)$, h_T is an experimentally determined heat transfer coefficient, and T_s is the temperature of the surroundings. Scaling (208) leads to

$$-\frac{\kappa\Delta t}{L}\frac{\partial\bar{T}}{\partial\bar{n}} = h_T(\Delta T\bar{T} + T_0 - T_s),$$

and further to

$$\frac{\partial\bar{T}}{\partial\bar{n}} = \frac{h_T L}{\kappa}(\bar{T} + \frac{T_s - T_0}{\Delta T}) = \mathrm{Nu}(\bar{T} - \bar{T}_s),$$

where the Nusselt number is defined by

$$\mathrm{Nu} = \frac{h_T L}{\kappa},$$

and $\bar{T}_s$ is simply the dimensionless surrounding temperature,

$$\bar{T}_s = \frac{T_s - T_0}{\Delta T}\,.$$

Heat transfer is a huge engineering field with lots of experimental investigations that are summarized by curves relating various dimensionless numbers such as Gr, Pr, and Nu.

10 The bidomain model in electrophysiology

The mechanical functioning of the heart is crucially dependent on correct electric signal propagation through the heart tissue. A widely used mathematical model for the electric signal propagation is the bidomain equations:

$$\chi C_m \frac{\partial v}{\partial t} = \nabla\cdot(M_i\nabla v) + \nabla\cdot(M_i\nabla u_e) - \chi I_{\text{ion}} - \chi I_{\text{app}}, \tag{209}$$

$$0 = \nabla\cdot(M_i\nabla v) + \nabla\cdot((M_i + M_e)\nabla u_e)\,. \tag{210}$$

These PDEs are posed in a spatial domain H for $t \in (0,T]$. The boundary conditions are of Neumann type, and we drop these from the discussion. The initial condition is typically $u_e = v = 0$.

The symbols in these PDEs have the following meaning: u_e is the extracellular electric potential, v is the transmembrane potential (difference between the extracellular and intracellular potential), C_m is the capacitance of the cell membrane, χ is a membrane area to cell volume ratio, M_i is a electric conductivity tensor for the intracellular space, M_e is a electric conductivity tensor for the extracellular space, I_{ion} is the ionic current across the cell membrane, and I_{app} is an externally applied current.

The PDE system is driven by $I_{\text{ion}} + I_{\text{app}}$, and models for these source terms consist of a system of ODEs at each point in the domain. The simplest relevant ODE system is the FitzHugh-Nagumo model:

$$C_m\frac{dv}{dt} = -I_{\text{ion}} - I_{\text{app}}, \tag{211}$$

$$I_{\text{ion}} = A(v - v_r)(v - v_m)(v - v_p) + Bw, \tag{212}$$

$$\frac{dw}{dt} = -c_1 w + c_2 v + c_3, \tag{213}$$

where A, v_m, B, c_1, c_2, and c_3 are specified constants. **hpl 16**: Keyner and Sneyd have a constant in the equation for w, in both the unscaled and scaled versions, while The Bible has not.

More complicated ODE systems for the cell dynamics may have up to a hundred unknowns and a correspondingly large collection of parameters.

Dimensionless independent variables are introduced by

$$\bar{x} = \frac{x}{L}, \quad \bar{y} = \frac{y}{L}, \quad \bar{z} = \frac{z}{L}, \quad \bar{t} = \frac{t}{t_c},$$

where L is the characteristic length scale, and t_c is the characteristic time scale. Dimensionless dependent variables are expressed as

$$\bar{v} = \frac{v - v_r}{v_p - v_r}, \quad \bar{u} = \frac{u_e}{u_c}, \quad \bar{w} = \frac{w}{w_c}\,.$$

Here, v_r is the resting potential, and v_p is the peak potential. The scaling of v ensures $\bar{v} \in [0,1]$. We introduce the symbol $\Delta v = v_p - v_r$ to save space in the formulas: $\bar{v} = (v - v_r)/\Delta v$. The scale for u_e is u_c, and w_c is a scale for w, both to be determined either from simplicity of the equations or from available analysis of their magnitudes.

The variable tensor coefficients M_i and M_e depend on the spatial coordinates and are also scaled:

$$\bar{M}_i = \frac{M_i}{M_c}, \quad \bar{M}_e = \frac{M_e}{M_c}\,.$$

For simplicity, we have chosen a common scale M_c, but the two tensors might employ difference scales. We may typically choose M_c as a norm of $M_i + M_e$, e.g., the maximum value.

Inserting the dimensionless variables in the equations, we achieve the system

$$\begin{aligned}
\frac{\Delta v}{t_c}\chi C_m\frac{\partial \bar{v}}{\partial \bar{t}} &= \frac{M_c\Delta v}{L^2}\nabla\cdot(\bar{M}_i\bar{\nabla}\bar{v}) + \frac{M_c u_c}{L^2}\nabla\cdot(\bar{M}_i\bar{\nabla}\bar{u}) - \\
&\quad \chi A\Delta v\bar{v}(v_r - v_m + \Delta v\bar{v})(v_r - v_p + \Delta v\bar{v}) - \\
&\quad Bw_c\bar{w} - \chi I_{\text{app}}, \\
0 &= \frac{M_c\Delta v}{L^2}\bar{\nabla}\cdot(\bar{M}_i\bar{\nabla}\bar{v}) + \frac{M_c u_c}{L^2}\nabla\cdot((\bar{M}_i + \bar{M}_e)\bar{\nabla}\bar{u}), \\
\frac{\Delta v}{t_c}C_m\frac{d\bar{v}}{d\bar{t}} &= -A\Delta v\bar{v}(v_r - v_m + \Delta v\bar{v})(v_r - v_p + \Delta v\bar{v}) - \\
&\quad Bw_c\bar{w} - I_{\text{app}}, \\
\frac{w_c}{t_c}\frac{d\bar{w}}{d\bar{t}} &= -c_1 w_c\bar{w} + c_2(v_r + \Delta v\bar{v}) + c_3\,.
\end{aligned}$$

Multiplying the equations by appropriate factors leads to equations with dimensionless terms only:

$$\begin{aligned}
\frac{\partial \bar{v}}{\partial \bar{t}} &= \frac{t_c M_c}{\chi C_m L^2}\nabla\cdot(\bar{M}_i\bar{\nabla}\bar{v}) + \frac{t_c M_c u_c}{\Delta v\chi C_m L^2}\nabla\cdot(\bar{M}_i\bar{\nabla}\bar{u}) - \\
&\quad \frac{t_c A}{C_m\Delta v}\Delta v\bar{v}(v_r - v_m + \Delta v\bar{v})\Delta v(\bar{v} - 1) - \\
&\quad \frac{t_c Bw_c}{\chi C_m\Delta v}\bar{w} - \frac{t_c I_{\text{app}}}{C_m\Delta v}, \\
0 &= \bar{\nabla}\cdot(\bar{M}_i\bar{\nabla}\bar{v}) + \frac{u_c}{\Delta v}\nabla\cdot((\bar{M}_i + \bar{M}_e)\bar{\nabla}\bar{u}), \\
\frac{d\bar{v}}{d\bar{t}} &= -\frac{t_c A}{C_m\Delta v}\Delta v\bar{v}(v_r - v_m + \Delta v\bar{v})\Delta v(\bar{v} - 1) - \\
&\quad \frac{t_c Bw_c}{C_m\Delta v}\bar{w} - \frac{t_c}{C_m\Delta v}I_{\text{app}}, \\
\frac{d\bar{w}}{d\bar{t}} &= -t_c c_1\bar{w} + \frac{t_c c_2}{w_c}(v_r + \Delta v\bar{v}) + \frac{t_c c_3}{w_c}\,.
\end{aligned}$$

The length scale is given as the size of the domain, in this case the diameter of the heart. The time scale is less obvious to choose. The PDEs are of diffusion nature, and the relevant time scale is related to the time it takes to diffuse the signal. On the other hand, the ODE system, in general, may feature many different time scales. Which time scale to choose for t_c depends

on whether one is interested in fast time scales for some components in the ODE system or in the (anticipated) slower diffusion scale of the PDEs. For now we choose the time scale based on diffusion. From previous examples in Section 5.1, we therefore set the coefficient in front of the diffusion term to unity, here

$$\frac{t_c M_c}{\chi C_m L^2} = 1 \quad \Rightarrow \quad t_c = \frac{\chi C_m L^2}{M_c}\,.$$

A natural dimensionless variable then arises from the second diffusion term:

$$\alpha = \frac{u_c}{\Delta v}\,.$$

The choice $u_c = \Delta v$ will of course remove the need for this dimensionless variable, but we include the freedom to have u_c as some specified characteristic size of u_e.

The final dimensionless system becomes

$$\frac{\partial \bar{v}}{\partial \bar{t}} = \nabla \cdot (\bar{M}_i \bar{\nabla} \bar{v}) + \alpha \nabla \cdot (\bar{M}_i \bar{\nabla} \bar{u}) - \tag{214}$$

$$\beta \bar{v}(J_1 + \bar{v})(\bar{v} - 1) - \gamma \bar{w} - \delta, \tag{215}$$

$$0 = \bar{\nabla} \cdot (\bar{M}_i \bar{\nabla} \bar{v}) + \alpha \nabla \cdot ((\bar{M}_i + \bar{M}_e) \bar{\nabla} \bar{u}), \tag{216}$$

$$\frac{d\bar{v}}{d\bar{t}} = -\beta \bar{v}(J_1 + \bar{v})(\bar{v} - 1) - \gamma \bar{w} - \delta, \tag{217}$$

$$\frac{d\bar{w}}{d\bar{t}} = -K_1 \bar{w} + K_2 \bar{v} + K_3\,. \tag{218}$$

There are eight dimensionless variables in the above equations:

$$\alpha = \frac{u_c}{\Delta v}, \tag{219}$$

$$\beta = \frac{\chi A L^2 \Delta v^2}{M_c}, \tag{220}$$

$$\gamma = \frac{B w_c L^2}{M_c \Delta v}, \tag{221}$$

$$\delta = \frac{\chi L^2 I_{\text{app}}}{M_c \Delta v}, \tag{222}$$

$$J_1 = \frac{v_r - v_m}{\Delta v}, \tag{223}$$

$$J_2 = \frac{v_r}{\Delta v}, \tag{224}$$

$$K_1 = \frac{\chi C_m L^2 c_1}{M_c}, \tag{225}$$

$$K_2 = \frac{\chi C_m L^2 \Delta v^2 c_2}{M_c w_c}, \tag{226}$$

$$K_3 = \frac{\chi C_m L^2 (c_3 + c_2 v_r \Delta v)}{M_c w_c}\,. \tag{227}$$

Since w is just a help variable without any physiological interpretation, we can choose its scale w_c freely to simplify the expressions. For example, we can fit w_c to make $K_2 = 1$, $K_3 = 1$, or $\gamma = 1$.

Some of these dimensionless numbers have straightforward interpretations: α is the ratio of the span in the two electric potentials; δ is ratio of the source term and the time-derivative term of v, or the source term and the diffusion term in v. The rest have fixed values: J_1 and J_2 have well-established values from measurements, while β, γ, K_1, K_2, and K_3 contain empirically fitted constants with known values, combined with known scales and other known physical parameters.

hpl 17: It is unclear what the gain really is, besides having primary variables of unit size. Since there are so many empirical constants, the number of dimensionlesss variables blows up, and instead of gettting more insight into the dynamics through these variables, they are just fixed numbers.

11 Two-phase porous media flow

Exercises: single-phase flow, with and without thermal effects.

12 The Euler equations of gas dynamics

13 Exercises

Exercise 1: Perform unit conversion

Density (mass per volume: $[\mathrm{ML}^{-3}]$) of water is given as 1.05 ounce per fluid ounce. Use the `PhysicalQuantity` object to convert to $\mathrm{kg\,m}^{-3}$.
Filename: `density_conversion`.

Problem 2: Scale a simple formula

The height y of a body thrown up in the air is given by

$$y = v_0 t - \frac{1}{2} g t^2,$$

where t is time, v_0 is the initial velocity of the body at $t = 0$, and g is the acceleration of gravity. Scale this formula. Use two choices of the characteristic time: the time it takes to reach the maximum y value and the time it takes to return to $y = 0$.
Filename: `scaled_vertical_motion`.

Problem 3: Scale a nonlinear ODE

The velocity $v(t)$ of a body moving vertically through a fluid in the gravity field, with fluid drag and buoyancy, is governed by ODE

$$mv' = -\frac{1}{2} C_D \varrho A |v| v - mg + \varrho V g, \quad v(0) = v_0,$$

where t is time, m is the mass of the body, C_D is a drag coefficient, ϱ is the density of the fluid, A is the cross-sectional area perpendicular to the motion, g is the acceleration of gravity, and V is the volume of the body. Scale this ODE.
Filename: `scaled_vertical_motion_with_drag`.

Exercise 4: Implement a scaled model with jump

Make software for the problem in Section 2.4 so that you can produce Figure 4.

Hint. Follow the ideas for software in Section 2.3: use the `decay_vc.py`[21] module as computational engine and modify the `falling_body.py`[22] code.
Filename: `decay_jump`.

Exercise 5: Implement a scaled model for cooling

Make software for the unscaled problem (17) where T_s can be a function of time. Use this implementation to compute the solution of the scaled problem (24).

Hint. You may use the general software `decay_vc.py`[23] for computing with the cooling model. See Section 2.3 for more ideas.
Filename: `decay_cooling1`.

Problem 6: Scale variable coefficients

The goal of this exercise is to scale the problem $u'(t) = -a(t)u(t) + b(t)$, $u(0) = I$, when

$$a(t) = \begin{cases} Q, & t < s, \\ Q - A, & t \geq s, \end{cases} \qquad b = \begin{cases} \gamma t, & t < s, \\ 0, & t \geq s, \end{cases}$$

Here, $Q, A, \gamma > 0$. Filename: `decay_varcoeff`.

Exercise 7: Alternative scalings of a cooling model

Scale the model (17), with T_s given as in (23), using two alternative scalings of T: (21) and the simpler $\bar{T} = T/T_0$. Does the type of scaling impact how many dimensionless parameters we end up with? Filename: `decay_cooling2`.

Exercise 8: Alternative scalings of a cooling model

Implement the scaled model (30) and produce a plot with curves corresponding to various values of α and p to summarize how $\bar{u}(\bar{t})$ look like.

Hint. A centered Crank-Nicolson-style scheme for (30) can use an old time value for the nonlinear coefficient:

$$\frac{\bar{u}^{n+1} - \bar{u}^n}{\Delta t} = (1 - \alpha\bar{u}^n)^p \frac{1}{2}(\bar{u}^n + \bar{u}^{n+1}) \,.$$

Filename: `growth`.

[21] `http://tinyurl.com/nm5587k/decay/decay_vc.py`
[22] `http://tinyurl.com/nm5587k/scale/falling_body.py`
[23] `http://tinyurl.com/nm5587k/decay/decay_vc.py`

Exercise 9: Scale projectile motion

We have the following mathematical model for the motion of a projectile in two dimensions:

$$m\ddot{\boldsymbol{x}} + \frac{1}{2}C_D\varrho A|\dot{\boldsymbol{x}}|\dot{\boldsymbol{x}} = -mg\boldsymbol{j}, \quad \boldsymbol{x}(0) = \boldsymbol{0},\ \dot{\boldsymbol{x}}(0) = v_0\cos\theta\boldsymbol{i} + v_0\sin\theta\boldsymbol{j}\,.$$

Here, m is the mass of the projectile, $\boldsymbol{x} = x\boldsymbol{i} + y\boldsymbol{j}$ is the position vector of the projectile, $\boldsymbol{i}$ and $\boldsymbol{j}$ are unit vectors along the x and y axes, respectively, $\ddot{\boldsymbol{x}}$ and $\dot{\boldsymbol{x}}$ is the second- and first-order time derivative of $\boldsymbol{x}(t)$, C_D is a drag coefficient depending on the shape of the projectile (can be taken as 0.4 for a sphere), ϱ is the density of the air, A is the cross section area (can be taken as πR^2 for a sphere of radius R), g is gravity, v_0 is the initial velocity of the projectile in a direction that makes the angle θ with the ground.

a) Neglect the air resistance term proportional to $\dot{\boldsymbol{x}}$ and solve analytically for $\boldsymbol{x}(t)$.

b) Make the model for projectile motion with air resistance non-dimensional. Use the maximum height from the simplification in a) as length scale.

c) Make the model dimensionless again, but this time by demanding that the scaled initial velocity is unity in x direction.

d) A soccer ball has diameter $R = 11$ cm and mass 0.43 kg, the density of air is 1.2 kgm^{-3}, a soft kick has velocity 10 km/h, while a hard kick may have 120 km/h. Estimate the dimensionless parameter in the scaled problem for a soft and a hard kick with θ corresponding to 30 degrees. Solve the scaled differential equation for these values and plot the trajectory (y versus x) for the two cases.
Filename: `projectile`.

Problem 10: Scale a predator-pray model

The evolution of animal populations with a predator and a pray (e.g., lynx and hares or foxes and rabbits) can be described by the ODE problem

$$H' = H(a - bL), \tag{228}$$
$$L' = L(dH - c), \tag{229}$$
$$H(0) = H_0, \tag{230}$$
$$L(0) = L_0\,. \tag{231}$$

Here, H is the number of animals of the pray (say hares) and L is the corresponding measure of the predator population. There are six parameters: a, b, c, d, H_0, and L_0.

The terms has the following meanings:

- aH is the exponential population growth of H due to births and deaths and is governed by the access to nutrition,
- $-bHL$ is the loss of prays because they are eaten by predators,
- dHL is the increase of prays because they eat predators (but only a fraction of the eaten prays, bHL, contribute to population growth of the predator and therefore $d < b$),
- $-cL$ is the exponential decay in the predator population because of deaths (the increase is modeled by dHL).

Dimensionless independent and dependent variables are introduces as usual by

$$\bar{t} = \frac{t}{t_c}, \quad \bar{H} = \frac{H}{H_c}, \quad \bar{L} = \frac{L}{L_c},$$

where t_c, H_c, and L_c are scales to be determined. Inserted in the ODE problem we arrive at

$$\frac{H_0}{t_c}\frac{d\bar{H}}{d\bar{t}} = H_0\bar{H}(a - bH_0\bar{L}), \tag{232}$$

$$\frac{H_0}{t_c}\frac{d\bar{L}}{d\bar{t}} = H_0\bar{L}(dH_0\bar{H} - c), \tag{233}$$

$$H_c\bar{H}(0) = H_0, \tag{234}$$

$$L_c\bar{H}(0) = L_0\,. \tag{235}$$

a) Consider first an intuitive scaling of H and L where $H_c = H_0$ and $L_c = H_c$. That is, $\bar{H}$ starts out at unity and $\bar{L}$ starts out as the fraction L_0/H_0. Find a time scale and identify dimensionless parameters in the scaled ODE problem.

b) Try a different scaling where the aim is to adjust the scales such that the ODEs become as simple as possible, i.e, have as few dimensionless parameters as possible. Compare with the scaling in a).
Filename: `predator_pray`.

Problem 11: Find the period of sinusoidal signals

a) Plot the function

$$u(t) = A\sin(\omega t),$$

for $t \in [0, 8\pi/\omega]$. Choose ω and A.

b) The *period* P of u is the shortest distance between two peaks (where $u = A$). Show mathematically that

$$P = \frac{2\pi}{\omega}\,.$$

Frequently, P is also referred to as the *wave length* of u.

c) Plot the damped signal $u(t) = e^{-at}\sin(\omega t)$ over four periods of $sin(\omega t)$. Choose ω, A, and a.

d) What is the period of $u(t) = e^{-at}\sin(\omega t)$? We define the period P as the shortest distance between two peaks of the signal.

Hint. Use that $v = p\cos(\omega t) + q\sin(\omega t)$ can be rewritten as $v = B\cos(\omega t - \phi)$ with $B = \sqrt{p^2 + q^2}$ and $\phi = \tan^{-1}(p/q)$. Use such a rewrite of u' to find the peaks of u and then the period.
Filename: `sine_period`.

Remarks. The *frequency* is the number of up and down cycles in one unit time. Since there is one cycle in a period P, the frequency is $f = 1/P$, measured in Hz. The *angular frequency* ω is then $\omega = 2\pi/P = 2\pi f$.

Problem 12: Scale the pendulum equation

The equation for a so-called simple pendulum with a mass m at the end is

$$mL\ddot{\theta} + mg\sin\theta = 0, \tag{236}$$

where $\theta(t)$ is the angle with the vertical, L is the length of the pendulum, and g is the acceleration of gravity.

A physical pendulum with moment of inertia I is governed by a similar equation,

$$I\ddot{\theta} + mgL\sin\theta = 0\,. \tag{237}$$

Both equations have the initial conditions $\theta(0) = \Theta$ and $\theta'(0) = 0$ (start at rest).

Use θ as dimensionless unknown, find a proper time scale, and scale both differential equations.
Filename: `pendulum`.

Problem 13: Scale Duffing's equation

Duffing's equation is a vibration equation with linear and cubic spring terms:

$$mu'' + k_0 u + k_1 u^3 = 0, \quad u(0) = U_0,\ u'(0) = 0\,.$$

Scale this problem.
Filename: `Duffing_eq`.

Problem 14: Scale a stationary Couette flow

A fluid flows between two flat plates, with one plate at rest while the other moves with velocity U_0. This classical flow case is known as stationary Couette flow.

a) Directing the x axis in the flow direction and letting y be a coordinate perpendicular to the walls, one can assume that the velocity field simplifies to $\boldsymbol{u} = u(y)\boldsymbol{i}$. Show from the Navier-Stokes equations that the boundary-value problem for $u(y)$ is

$$u''(u) = 0, \quad u(0) = 0,\ u(H) = U_0\,.$$

We have here assumed at $y = 0$ corresponds to the plate at rest and that $y = H$ represents the plate that moves. There are no pressure gradients present in the flow.

b) Scale the problem in a) and show that the result has no physical parameters left in the model:

$$\frac{d^2\bar{u}}{d\bar{y}^2} = 0, \quad \bar{u}(0) = 0,\ \bar{u}(1) = 1\,.$$

c) We can compute $\bar{u}(\bar{y})$ from one numerical simulation (or a straightforward integration of the differential equation). Set up the formula that finds $u(y; H, u_0)$ from $\bar{u}(\bar{y})$ for any values of H and U_0.
Filename: `stationary_Couette`.

Remarks. The problem for u is a classical two-point boundary-value problem in applied mathematics and arises in a number of applications, where Couette flow is just one example. Heat conduction is another example: u is temperature, and the heat conduction equation for an insulated rod reduces to $u'' = 0$ under stationary conditions and no heat source. Held the end $x = 0$ at 0 degrees Celsius the other end $x = L$ at U_0 degrees Celsius, gives the same boundary conditions as in the above flow problem. The scaled problem is of course the same whether we have flow of fluid or heat.

Problem 15: Scale a starting Couette flow

A fluid is confined in a channel with two planar walls $z = 0$ and $z = H$. The fluid is at rest. At time $t = 0$ the upper wall is suddenly set in motion with a velocity $U\boldsymbol{i}$. We assume that the velocity is directed along the x axis: $\boldsymbol{u} = u(x, z, t)\boldsymbol{i}$. From the equation of continuity, $\nabla \cdot \boldsymbol{u} = 0$, we get that $\partial u/\partial x = 0$ such that $\boldsymbol{u} = u(z, t)\boldsymbol{i}$. The boundary conditions are $\boldsymbol{u} = 0$ at the lower wall $z = 0$ and $\boldsymbol{u} = U\boldsymbol{i}$ at the upper wall $z = H$. Assume that the pressure is constant everywhere and that there are no body forces.

a) Start with the incompressible Navier-Stokes equations and the assumption $\boldsymbol{u} = u(z, t)\boldsymbol{i}$. Derive an initial-boundary value problem for $u(z, t)$. Scale the problem.

b) Start with the dimensionless Navier-Stokes equations and use the assumption $\bar{\boldsymbol{u}} = \bar{u}(\bar{z}, \bar{t})\boldsymbol{i}$ to reduce the problem. The resulting equation now contains a Reynolds number, i.e., one more physical parameter than in a). Why is this an inferior approach to scaling the problem?

c) Can you construct a heat conduction problem that has the same solution $\bar{u}(\bar{z}, \bar{t})$ as in a)?

d) Describe how the scaled problem in this exercise can be solved by a program that solves the following diffusion problem with dimensions:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \alpha \frac{\partial^2 u}{\partial z} + f(x, t), \\ u(x, 0) &= I(x), \\ u(0, t) &= U_0(t), \\ u(L, t) &= U_L(t)\,. \end{aligned}$$

Filename: `starting_Couette`.

Exercise 16: Scale Couette flow with pressure gradient

Viscous fluid flow between two infinite flat plates $z = 0$ and $z = H$ is governed by

$$\mu u''(z) = -\beta \tag{238}$$

$$u(0) = 0, \tag{239}$$

$$u(H) = U_0\,. \tag{240}$$

Here, $u(z)$ is the fluid velocity in x direction (perpendicular to the z axis), μ is the dynamic viscosity of the fluid, β is a positive constant pressure gradient, and U_0 is the constant velocity of the upper plate $z = H$ in x direction.

a) Find the exact solution $u(z)$. Point out how β and U_0 influence the magnitude of u.

b) Scale the problem.
hpl 21: Could extend to time-dependent case, but this will involve three time scales...
Filename: `Couette_wpressure`.

Exercise 17: Suggestions...

Projects:

- Poisson problem, membrane, see Nayfeh chapter 1, stationary and time
- diffusion with oscillating $f(x,t)$, oscillating pressure in tube (channel first)
- Helmholtz
- two metal pieces in contact
- boundary layer fluid flow problem, with/without thermal effects
- channel viscous flow: with and without Reynolds number (bad to have an extra parameter!)
- Starting Couette flow and rod with fixed end temperatures
- Fisher's equation
- Vertical motion in the gravity field, two time scales, discuss
- sliding box, see `vib` exercise

References

[1] J. F. Douglas, J. M. Gasiorek, and J. A. Swaffield. *Fluid Mechanics*. Pitman, 1979.

[2] H. P. Langtangen. Finite difference methods for wave motion. *http://tinyurl.com/k3sdbuv/pub/wave*.

[3] H. P. Langtangen. Scientific software engineering; ODE case. *http://tinyurl.com/k3sdbuv/pub/softeng1*.

[4] H. P. Langtangen. Scientific software engineering; wave equation case. *http://tinyurl.com/k3sdbuv/pub/softeng2*.

[5] H. P. Langtangen and A. E. Johansen. The Parampool tutorial. *http://hplgit.github.io/parampool/doc/web/index.html*.

[6] H. P. Langtangen and A. E. Johansen. Using web frameworks for scientific applications. *http://hplgit.github.io/web4sciapps/doc/web/index.html*.

[7] J. D. Logan. *Applied Mathematics: A Contemporary Approach*. Wiley, 1987.

[8] D. J. Tritton. *Physical Fluid Dynamics*. Van Nostrand Reinhold, 1977.

Index